Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Другий (магістерський)
(рівень вищої освіти)

Комп'ютерне моделювання сенсорної системи мобільного робота у симуляторі WeBots
(тема)

Виконав:
здобувач _____ року навчання,
групи КТРСм-23-1
Буць Д.Є.
(прізвище, ініціали)
Спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Комп'ютерні та робототехнічні системи
(повна назва освітньої програми)
Керівник доц. Чала О.О.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри КІТАР _____ Невлюдов І. Ш.
(підпис) (прізвище, ініціали)

2024 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Автоматики та комп'ютеризованих технологій
Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
Рівень вищої освіти Другий (магістерський)
Спеціальність 174 Автоматизація та комп'ютерно-інтегровані технології
Тип програми освітньо-професійна
Освітня програма Комп'ютерні та робототехнічні системи
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри КІТАР___
(підпис)
«____»_____2024р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувачеві Буць Дмитру Євгеновичу
(прізвище, ім'я, по батькові)

1. Тема роботи Комп'ютерне моделювання сенсорної системи мобільного робота у симуляторі WeBots
Затверджена наказом по університету від 25.11.2024 № 1239 Ст
2. Термін подання студентом роботи до екзаменаційної комісії: 16.01.2025 року
3. Вихідні дані до роботи: Комп'ютерні та робототехнічні системи, симулятор WeBots, двоколісний робот, ПІД-регулятор, перевірка системи за критеріями стійкості.

4. Перелік питань, що потрібно опрацювати в роботі
4.1 Вступ
4.2 Аналіз сучасного стану систем моделювання у Індустрії 5.0
4.3 Вибір та обґрунтування основних елементів системи моделювання
4.4 Розроблення моделі мобільного робота та його сенсорної системи у середовищі WeBots
4.5 Проведення експериментальних досліджень сенсорної системи розробленого робота
4.6 Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайди у форматі Power Point у кількості 11 слайдів з розширенням .pptx

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Приміт-ка
1	Аналіз технічного завдання	27.11.2024	виконано
2	Опрацювання літератури за темою роботи.	27.11.2024 – 1.12.2024	виконано
3	Аналіз сучасного стану систем моделювання у Індустрії 5.0	2-12.2024 – 4.12.2024	виконано
4	Вибір та обґрунтування основних елементів системи моделювання	5.12.2024 – 9.12.2024	виконано
5	Розроблення моделі мобільного робота та його сенсорної системи у середовищі WeBots	10.12.2024 – 15.12.2024	виконано
6	Проведення експериментальних досліджень сенсорної системи розробленого робота	16.12.2024– 20.2024	виконано
7	Оформлення пояснювальної записки	20.12.2024 – 30.12.2024	виконано
8	Оформлення презентації	1.01.2025–	виконано
9	Подання роботи на рецензію	6.01.2025	виконано
10	Подання роботи на підпис зав. кафедри	10.01.2024	виконано
11	Подання атестаційної роботи в ЕК	16.01.2025	

Дата видачі завдання 25.11.2024

Здобувач _____ Буць Д.Є.
(підпис) (прізвище, ініціали)

Керівник роботи _____ доц. Чала О.О.
(підпис) (посада, прізвище, ініціали)

Я, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

31 січня 2024 року  Буць Д.Є.

РЕФЕРАТ

Кваліфікаційна робота містить: 107 с., 3 табл., 108 рис., 3 дод., 33 джерел.

КОМП'ЮТЕРНЕ МОДЕЛЮВАННЯ, WEBOTS, ПІД-РЕГУЛЯТОР, СЕНСОРНА СИСТЕМА, МОБІЛЬНИЙ РОБОТ, PYTHON.

Мета кваліфікаційної роботи – підвищення ефективності керування роботизованих систем за рахунок розроблення їх комп'ютерної моделі, шляхом розробки симуляції роботи робота та його сенсорної системи.

Об'єкт дослідження – системи обробки інформації мобільного робота. Предмет дослідження – параметри налаштування сенсорної системи.

В кваліфікаційній роботі розглянуто актуальні питання за темою, запропоновані рішення моделювання роботів у симуляторі WeBots, а також сенсорної системи, її налаштування та створення програмного забезпечення для керування комп'ютерною моделлю цього робота з використанням ІЧ-датчиків та ПІД-регулятора.

Результати роботи можна віднести до Цілей сталого розвитку 9 та 12: «Промисловість, інновації та інфраструктураа», «Відповідальне споживання та виробництво» відповідно.

Отримані результати можна використовувати на етапах проектування роботизованих систем, шляхом використання математичних моделей симуляції роботи робота та його сенсорної системи для тестування та наладки, а також в освітньому процесі при проведенні практичних чи лабораторних робіт за спеціальністю 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка.

Результати, що було отримано під час навчання та підготовки кваліфікаційної роботи висвітлено в статті.

ABSTRACT

The qualification work contains: 107 pages, 3 tables, 108 figures, 3 appendices, 33 sources.

COMPUTERN MODEL, WEBOTS, PID CONTROLLER, SENSOR SYSTEM, MOBILE ROBOT, PYTHON.

The goal of the qualification work is to enhance the control of robotic systems by creating a computer model through the simulation of robot operations and its sensor system.

The research examines the information processing systems of mobile robots.

This work addresses current challenges in the field, proposes methods for modeling robots in the WeBots simulator, and explores the setup of sensor systems. It also includes the development of software for controlling the robot’s computer model using infrared sensors and a PID controller.

The results of this work align with Sustainable Development Goals 9 and 12: "Industry, Innovation, and Infrastructure" and "Responsible Consumption and Production," respectively.

The outcomes can be applied during the design stages of robotic systems by utilizing mathematical simulation models of robot operations and sensor systems for testing and debugging. Additionally, these results can be used in the educational process during practical or laboratory work within the specialty 174 "Automation, Computer-Integrated Technologies, and Robotics".

The findings obtained during the study and preparation of this thesis published in a academic article.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМНІВ

Індустрія 5.0 – 5 промислова революція;

ІЧ – інфрачервоний;

ПІД-регулятор – пропорційно-інтегрально-диференційний регулятор;

API – інтерфейс програмування застосунків;

AR – доповнена реальність;

ROS – операційна система роботів;

VR – віртуальна реальність.

ВСТУП

Індустрія 5.0 представляє наступний етап промислової еволюції, зосереджуючись на синергії між людьми та машинами. На відміну від Індустрії 4.0, яка наголошує на автоматизації та обміні даними, Індустрія 5.0 має на меті запровадити людино-центрований підхід до виробництва. Ця зміна парадигми надає пріоритет людській творчості, налаштуванню та співпраці з передовими технологіями, зокрема робототехнікою та штучним інтелектом.

Моделювання роботів відіграє вирішальну роль у цю нову індустріальну еру. Він передбачає створення віртуальних моделей роботизованих систем для перевірки та оптимізації їх продуктивності в змодельованому середовищі перед розгортанням їх у реальних програмах. Цей підхід має кілька переваг:

– зменшення ризику: шляхом імітації робототехнічних операцій потенційні проблеми можна виявити та вирішити без ризику пошкодження фактичного обладнання чи шкоди працівникам;

– економічна ефективність: моделювання допомагає точно налаштувати процеси та зменшити потребу у фізичних прототипах, заощаджуючи час і ресурси;

– покращена співпраця: моделювання може моделювати взаємодію людини та робота, забезпечуючи безпечну та ефективну роботу колаборативних роботів (коботів) разом із людьми-операторами.

Таким чином, Індустрія 5.0 і моделювання роботів разом прокладають шлях до більш інтегрованого, ефективного та орієнтованого на людину промислового середовища. Ця комбінація не тільки підвищує продуктивність, але й забезпечує відповідність технологічних досягнень людським цінностям і потребам.

Мета кваліфікаційної роботи – підвищення ефективності керування роботизованих систем за рахунок розроблення їх комп’ютерної моделі, шляхом розробки симуляції роботи робота та його сенсорної системи.

Об’єкт дослідження – системи обробки інформації мобільного робота.

Предмет дослідження – параметри налаштування сенсорної системи.

В ході виконання кваліфікаційної роботи потрібно вирішити наступні задачі:

– провести аналіз сучасного стану систем моделювання у Індустрії 5.0;

– провести порівняння різних середовищ моделювання роботи роботів;

– обрати середовище моделювання для робота і його сенсорної системи;

– створити світ, в якому буде працювати віртуальна модель робота;

– створити комп’ютерну модель робота та його сенсорної системи;

– розробити програми керування для робота на основі даних з його сенсорної системи;

– провести експериментальні дослідження з розробленим роботом та його сенсорною системою;

– оформити пояснювальну записку кваліфікаційної роботи за допомогою [1] та [2].

Отримані результати можна використовувати на етапах проектування роботизованих систем, шляхом використання математичних моделей симуляції роботи робота та його сенсорної системи для тестування та ладки, а також в освітньому процесі при проведенні практичних чи лабораторних робіт за спеціальністю 174 Автоматизація, комп’ютерно-інтегровані технології та робототехніка.

Результати роботи можна віднести до Цілей сталого розвитку 9 та 12: «Промисловість, інновації та інфраструктура», «Відповідальне споживання та виробництво» відповідно.

Результати, що було отримано під час навчання та підготовки кваліфікаційної роботи висвітлено в статті [3]: Buts D. Signals Collisions Detection In Wireless Networks / D. Buts, O. Chala, S. Maksymova // Journal of Universal Science Research. – 2023. – № 1(11). – P. 156–168. (Додаток А).

1 АНАЛІЗ СУЧАСНОГО СТАНУ СИСТЕМ МОДЕЛЮВАННЯ У ІНДУСТРІЇ 5.0

1.1 Індустрія 5.0 та роль моделювання у ній

Індустрія 5.0 охоплює кілька ключових аспектів, які відрізняють її від попередніх промислових революцій (рис. 1.1). Ось деякі з основних компонентів [4-5]:

Рисунок 1.1 – Індустрія 5.0

– людиноцентричний підхід: Індустрія 5.0 робить сильний акцент на ролі людини у виробничому процесі. Він спрямований на посилення людської творчості та досвіду шляхом інтеграції їх із передовими технологіями. Цей підхід гарантує, що технологія служить для розширення людських можливостей, а не замінює їх;

– сталість: є основним принципом Індустрії 5.0. Це передбачає оптимізацію використання ресурсів, зменшення відходів і мінімізацію впливу на навколишнє середовище. Сталі практики інтегровані в усі аспекти виробництва, від проектування до утилізації;

– стійкість: побудова стійких систем, здатних адаптуватися до змін і відновлюватися після збоїв, має вирішальне значення. Це включає розробку гнучких виробничих процесів і надійних ланцюжків поставок, які можуть протистояти різноманітним викликам;

– гіперіндивідуалізація: Індустрія 5.0 дає змогу виробляти продукцію з високим ступенем індивідуальності, адаптовану до індивідуальних потреб клієнтів. Передові технології, як-от штучний інтелект і робототехніка, забезпечують більшу гнучкість і точність у виробництві, що дає змогу виробляти унікальні вироби в масштабі;

– етична та соціальна відповідальність: велика увага приділяється етичним практикам та соціальній відповідальності. Це включає забезпечення справедливої трудової практики, сприяння різноманітності та інклюзії, а також врахування ширшого соціального впливу промислової діяльності;

– передові технології: інтеграція передових технологій, таких як AI, IoT і робототехніка, є фундаментальною для Індустрії 5.0. Ці технології разом створюють розумні взаємопов'язані системи, які підвищують ефективність і продуктивність.

Напрямок сталого розвитку промисловості 5.0, який стає все більш важливим у сучасному промисловому ландшафті. Серед його основних напрямків є [6-7]:

– ефективність використання ресурсів: Індустрія 5.0 наголошує на ефективному використанні ресурсів для мінімізації відходів і зменшення впливу на навколишнє середовище. Це передбачає оптимізацію виробничих процесів, переробку матеріалів і використання відновлюваних джерел енергії. Моделювання допомагає змоделювати ці процеси для визначення найбільш стійких практик (рис. 1.2);

Рисунок 1.2 – Ефективне використання ресурсів

– циркулярна економіка: ключовим компонентом сталого розвитку є циркулярна економіка, де продукти призначені для повторного використання, реконструкції та переробки. Це зменшує потребу в сировині та мінімізує відходи. Симуляції можуть моделювати життєвий цикл продуктів, щоб переконатися, що вони розроблені з урахуванням екологічності;

– управління енергією: ефективне управління енергією має вирішальне значення для зменшення вуглецевого сліду промислових операцій. Моделювання може оптимізувати використання енергії шляхом моделювання різних сценаріїв і визначення найбільш енергоефективних процесів. Це включає інтеграцію відновлюваних джерел енергії, таких як сонячна та вітрова енергія (рис. 1.3);

Рисунок 1.3 – Ефективне управління енергією

– стійкі ланцюги поставок: Індустрія 5.0 сприяє розвитку стійких ланцюгів поставок, які є стійкими та адаптованими. Це передбачає відповідальне постачання матеріалів, зменшення викидів під час

транспортування та забезпечення чесної трудової практики. Симуляції можуть допомогти у плануванні та оптимізації операцій ланцюга постачання для підвищення стійкості;

– моніторинг навколишнього середовища: вдосконалені датчики та пристрої IoT використовуються для моніторингу умов навколишнього середовища в режимі реального часу. Ці дані можна використати для моделювання для прогнозування та пом’якшення впливу промислової діяльності на навколишнє середовище. Наприклад, моніторинг якості повітря та води може допомогти у вживанні профілактичних заходів для запобігання забрудненню.

Індустрія 5.0 представляє наступну еволюцію промислового розвитку, зосереджену на співпраці між людьми та машинами для створення більш персоналізованих та стійких виробничих процесів. На відміну від Індустрії 4.0, яка наголошує на автоматизації та обміні даними, Індустрія 5.0 спрямована на інтеграцію людської творчості та досвіду з передовими технологіями [8].

Моделювання відіграє вирішальну роль в Індустрії 5.0, надаючи віртуальне середовище для моделювання та тестування виробничих систем. Ключові напрямки та переваги моделювання є такими [8] (рис. 1.4):

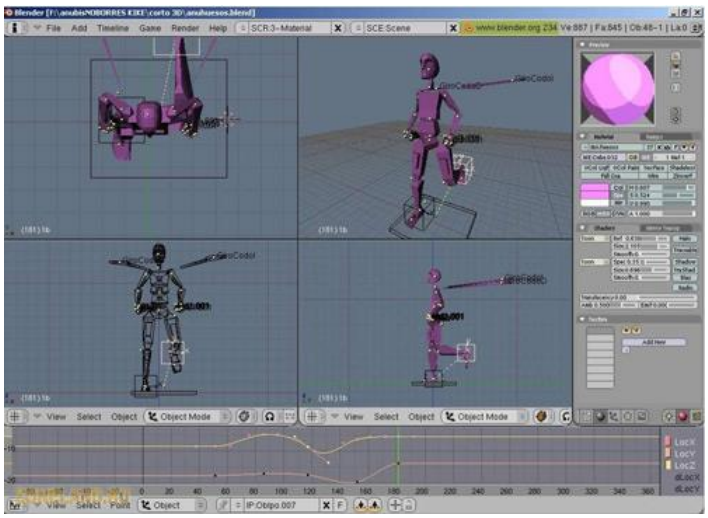

Рисунок 1.4 – Ефективність моделювання

– експерименти без ризику: моделювання дозволяє тестувати нові ідеї та процеси без ризику зриву фактичного виробництва. Ця віртуальна пісочниця дозволяє компаніям досліджувати альтернативи та оптимізувати роботу;

– цифрові двійники: це віртуальні копії фізичних систем, які можна використовувати для моніторингу та імітації операцій у реальному часі. Цифрові близнюки допомагають передбачати результати, визначати потенційні проблеми та покращувати процес прийняття рішень;

– співпраця людини та машини: інструменти моделювання дозволяють проектувати та тестувати системи, у яких люди та роботи працюють разом. Це гарантує, що обидва можуть працювати безпечно та ефективно в спільному середовищі;

– оптимізація та ефективність: моделюючи різні сценарії, компанії можуть визначити найефективніші процеси, зменшити відходи та підвищити продуктивність;

– навчання та розвиток навичок: симуляції забезпечують безпечний простір для навчання та відпрацювання нових навичок працівникам, гарантуючи, що вони добре підготовлені до роботи з передовими технологіями.

Співпраця людини і робота в Індустрії 5.0 є особливо важливим чинником для таких напрямків як [9]:

– колаборативні роботи (коботи) (рис. 1.5): коботи розроблені для роботи разом з людьми, підвищуючи продуктивність і безпеку. На відміну від традиційних промислових роботів, яких часто тримають у клітці з міркувань безпеки, коботи можуть працювати в безпосередній близькості від працівників. Вони оснащені вдосконаленими датчиками та штучним інтелектом для виявлення присутності людей і уникнення зіткнень;

– розширені заходи безпеки: моделювання відіграє вирішальну роль у розробці та тестуванні протоколів безпеки для взаємодії людини з роботом. Моделюючи різні сценарії, інженери можуть визначити потенційні небезпеки

та розробити стратегії їх пом’якшення. Це забезпечує більш безпечне робоче середовище, де люди та роботи можуть гармонійно співіснувати;

Рисунок 1.5 – Приклад кобота

– персоналізоване виробництво: Індустрія 5.0 наголошує на налаштуваннях і персоналізації у виробництві. Роботів можна запрограмувати на виконання складних різноманітних завдань, які потребують високого ступеня точності. Симуляції допомагають точно налаштувати ці процеси, забезпечуючи адаптацію роботів до різних специфікацій продукту та вимог клієнтів;

– підвищення навичок: роботи можуть розширювати людські здібності, дозволяючи працівникам виконувати завдання, які інакше були б надто складними чи небезпечними. Наприклад, на складальних лініях роботи можуть виконувати підйом важких або точні завдання, тоді як люди зосереджуються на більш складних і творчих аспектах виробництва. Симуляції допомагають розробити ці робочі процеси для максимальної ефективності та безпеки;

– навчання та розвиток: моделювання та симуляція [10] процесів забезпечують безпечне середовище для навчання працівників взаємодії з роботами. Симуляції віртуальної реальності (VR) і доповненої реальності (AR) можуть відтворювати сценарії реального світу, допомагаючи працівникам

отримати практичний досвід без жодного ризику. Це прискорює процес навчання та гарантує, що працівники добре підготовлені до роботи з передовими роботизованими системами.

1.2 Симуляція та моделювання у епоху Індустрії 5.0

Моделювання є важливим інструментом в різних галузях, включаючи Індустрію 5.0 [11].

Види моделювання розподіляються на наступні типи:

– стохастичне моделювання: використовує випадкові входи та виходи для моделювання систем із властивою випадковістю, наприклад, моделювання методом Монте-Карло для оцінки фінансового ризику;

– детерміноване моделювання: моделює системи з передбачуваною поведінкою, де ті самі вхідні дані завжди дають однакові виходи, наприклад, моделювання виробничого процесу з фіксованими параметрами;

– динамічне моделювання: моделює системи, які змінюються з часом, приклад: імітація поширення хвороби з часом;

– симуляція дискретних подій: фокусується на системах, де зміни відбуваються в окремі моменти часу, наприклад, імітація прибуття клієнтів і обслуговування в черзі;

– статичне моделювання: моделює системи в певний момент часу без урахування еволюції в часі, наприклад: аналіз планування виробничого цеху;

– безперервне моделювання: моделює системи з постійними змінами з часом, наприклад, моделювання змін температури в хімічному реакторі.

Також є такі види моделювання як [12]:

– агентне моделювання: моделює дії та взаємодію окремих агентів у системі, наприклад, моделювання поведінки споживачів на ринку;

– динаміка системи: використовує цикли зворотного зв’язку та часові затримки для моделювання складних систем, приклад: моделювання динаміки ланцюгів поставок;

– моделювання методом Монте-Карло: використовує випадкову вибірку, щоб зрозуміти вплив ризику та невизначеності в моделях, наприклад, фінансове прогнозування та управління ризиками;

– цифрові близнюки: створює віртуальну копію фізичної системи для моніторингу та моделювання в реальному часі, наприклад, моніторинг та оптимізація роботи вітрової турбіни;

– аналіз кінцевих елементів: використовує чисельні методи для прогнозування того, як продукт реагує на реальні сили, вібрацію, тепло та інші фізичні ефекти, наприклад, імітація структурної цілісності мосту.

Види моделювання показані на рис. 1.6.

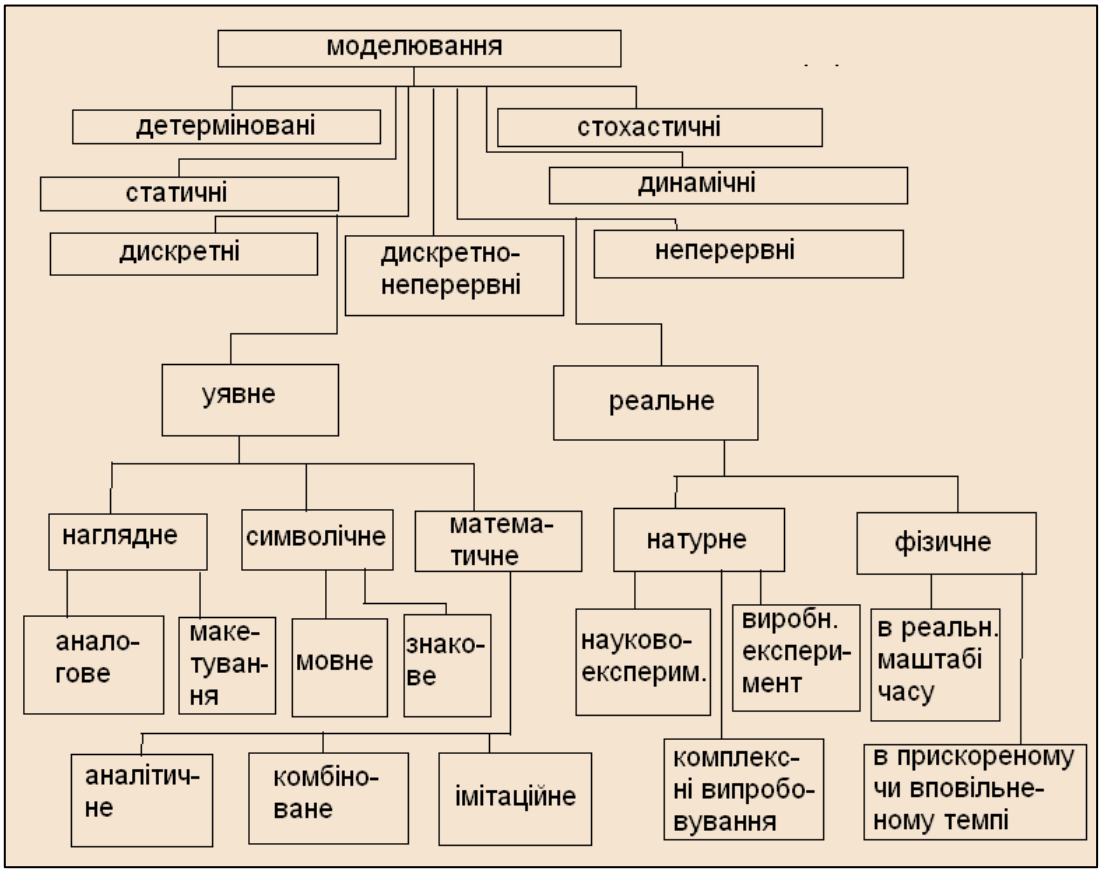

Рисунок 1.6 – Види моделювання

1.3 Комп’ютерне моделювання та його роль у Індустрії 5.0

Комп’ютерне моделювання передбачає використання комп’ютера для моделювання поведінки реальної чи гіпотетичної системи. Цей процес дозволяє дослідникам та інженерам вивчати, як система працює в різних умовах, не випробовуючи її фізично [13].

Комп’ютерне моделювання використовує математичні моделі для повторення динамічних реакцій однієї системи за допомогою поведінки іншої системи, змодельованої за нею. Також передбачає запуск цих моделей на комп’ютері для прогнозування результатів і поведінки системи, що вивчається (рис. 1.7).

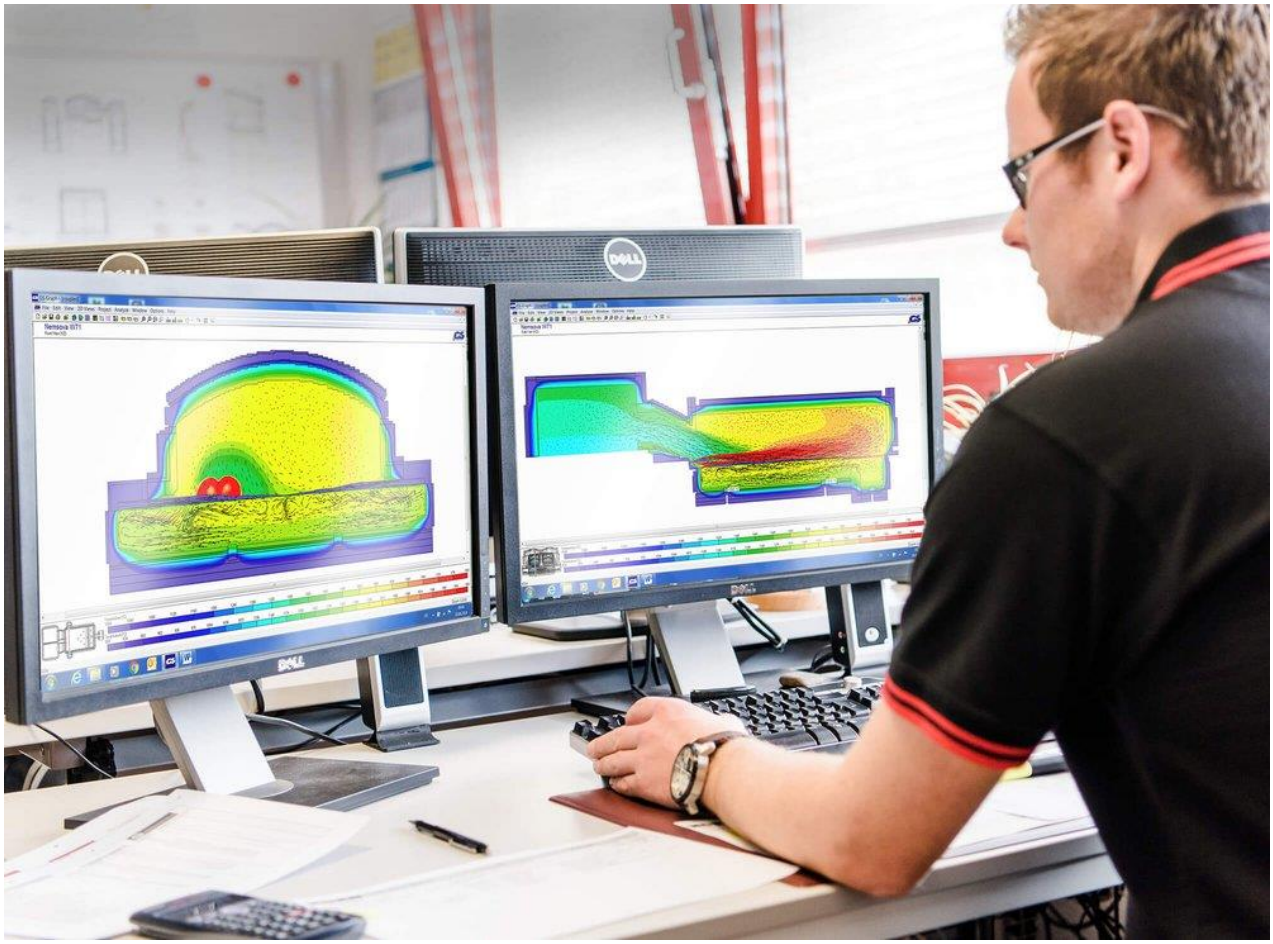

Рисунок 1.7 – Приклад комп’ютерного моделювання

Комп’ютерне моделювання використовується в різних сферах, зокрема:

– інженерна справа: для перевірки структурної цілісності будівель і мостів;

– робототехніка: розроблювати роботів та алгоритми їх керування;

– метеорологія: прогнозувати погодні умови та зміни клімату;

– охорона здоров’я: щоб імітувати поширення хвороб і вплив лікування;

– автомобільна промисловість: проектувати та тестувати нові моделі транспортних засобів;

– аерокосмічна: для моделювання космічних місій і поведінки космічних кораблів.

Перевагами використання є:

– зменшення ризику: дозволяє проводити тестування без ризику реальних наслідків;

– економічна ефективність: зменшує потребу у фізичних прототипах і експериментах;

– швидкість: прискорює процес тестування, швидко запускаючи кілька сценаріїв;

– статистика: надає детальну інформацію про поведінку системи та потенційні покращення.

Комп’ютерне моделювання відіграє ключову роль в Industry 5.0, забезпечуючи інтеграцію передових технологій із підходами, орієнтованими на людину, а саме таким чином [14]:

– віртуальне прототипування та тестування – комп’ютерне моделювання дозволяє створювати та тестувати віртуальні прототипи перед фізичним виробництвом. Перевагами є зменшення вартості і часу, пов’язаних з фізичним прототипом, і допомагає виявити потенційні недоліки конструкції на ранніх стадіях процесу розробки;

– цифрові двійники – це віртуальні копії фізичних систем, які постійно оновлюються даними в реальному часі. Перевагами є: вони забезпечують моніторинг у реальному часі, прогнозне технічне обслуговування та

оптимізацію промислових процесів. Наприклад, цифровий двійник виробничого підприємства може допомогти контролювати стан обладнання та прогнозувати збої;

– оптимізація процесу – симуляція моделює та аналізує виробничі процеси для виявлення неефективності та оптимізації операцій. Переваги: це сприяє підвищенню продуктивності, зменшенню відходів і покращенню контролю якості. Наприклад, імітація різних виробничих сценаріїв може допомогти знайти найбільш ефективний робочий процес;

– співпраця людини і робота – симуляції використовуються для розробки та тестування спільних завдань між людьми та роботами. Переваги: це забезпечує безпеку та ефективність у середовищах, де люди та роботи працюють разом. Наприклад, моделювання може допомогти в проектуванні складальних ліній, де роботи справляються з важким, а люди виконують точні завдання;

– управління ланцюгом поставок – імітаційні моделі використовуються для аналізу та оптимізації операцій ланцюга поставок. Переваги: це допомагає в управлінні складними глобальними ланцюжками поставок, зменшуючи витрати та покращуючи стійкість. Моделювання різних сценаріїв ланцюга постачання може виявити потенційні збої та оптимізувати логістику;

– енергетичний менеджмент – моделювання оптимізує використання енергії в промислових процесах. Переваги: це призводить до економії коштів і зменшення впливу на навколишнє середовище. Наприклад, моделювання моделей споживання енергії може допомогти в інтеграції відновлюваних джерел енергії та зменшенні загального споживання енергії.

Комп’ютерне моделювання є життєво важливим інструментом у сфері робототехніки, що пропонує численні переваги та застосування (рис. 1.8) [15]:

Рисунок 1.8 – Моделювання роботів

– дизайн і прототипування – комп’ютерне моделювання дозволяє інженерам створювати та тестувати віртуальні моделі роботів перед створенням фізичних прототипів – це зменшує витрати та час, пов’язані з фізичним прототипом, і допомагає виявляти недоліки дизайну на ранніх стадіях процесу розробки;

– тестування та валідація – симуляції дозволяють тестувати конструкції роботів, алгоритми керування та взаємодію з різними елементами у віртуальному середовищі без ризиків, що гарантує належну роботу роботів за різних умов і сценаріїв без ризику пошкодження фізичного обладнання;

– навчання та розвиток – симуляції забезпечують безпечне середовище для навчання операторів і розробників взаємодії з роботами – це прискорює процес навчання та гарантує, що користувачі добре підготовлені до роботи з реальними програмами;

– співпраця людини і робота – симуляції використовуються для розробки та тестування спільних завдань між людьми та роботами, а саме забезпечує безпеку та ефективність у середовищах, де люди та роботи працюють разом, наприклад на виробництві чи в установах охорони здоров’я;

– оптимізація роботизованих систем – симуляції допомагають оптимізувати продуктивність роботизованих систем шляхом моделювання різних сценаріїв і конфігурацій – це призводить до підвищення ефективності, зниження споживання енергії та покращення загальної продуктивності;

– моніторинг і контроль у реальному часі – цифрові близнюки, тип симуляції, забезпечують моніторинг і керування роботизованими системами в реальному часі, що дозволяє здійснювати прогнозне технічне обслуговування, коригування в режимі реального часу та покращувати процес прийняття рішень на основі реальних даних [16].

1.4 Переваги та недоліки комп’ютерного моделювання у завданнях робототехніки

Перевагами комп’ютерного моделювання в робототехніці є:

– економічна ефективність: зменшує потребу у фізичних прототипах і тестуванні, заощаджуючи гроші на матеріалах і робочій силі, наприклад, імітація рухів і взаємодій робота перед створенням фізичних моделей (рис. 1.9);

– зниження ризику: дозволяє проводити тестування у віртуальному середовищі, мінімізуючи ризик пошкодження дорогого обладнання або шкоди людям, наприклад, тестування нових робототехнічних алгоритмів у симуляції, щоб переконатися, що вони працюють правильно перед розгортанням (рис. 1.10) [17];

– швидкість і гнучкість: прискорює процес розробки, дозволяючи швидку ітерацію та тестування різних сценаріїв, а саме швидке модифікування

та тестування конструкцій роботів для пошуку найбільш ефективної конфігурації (рис. 1.11);

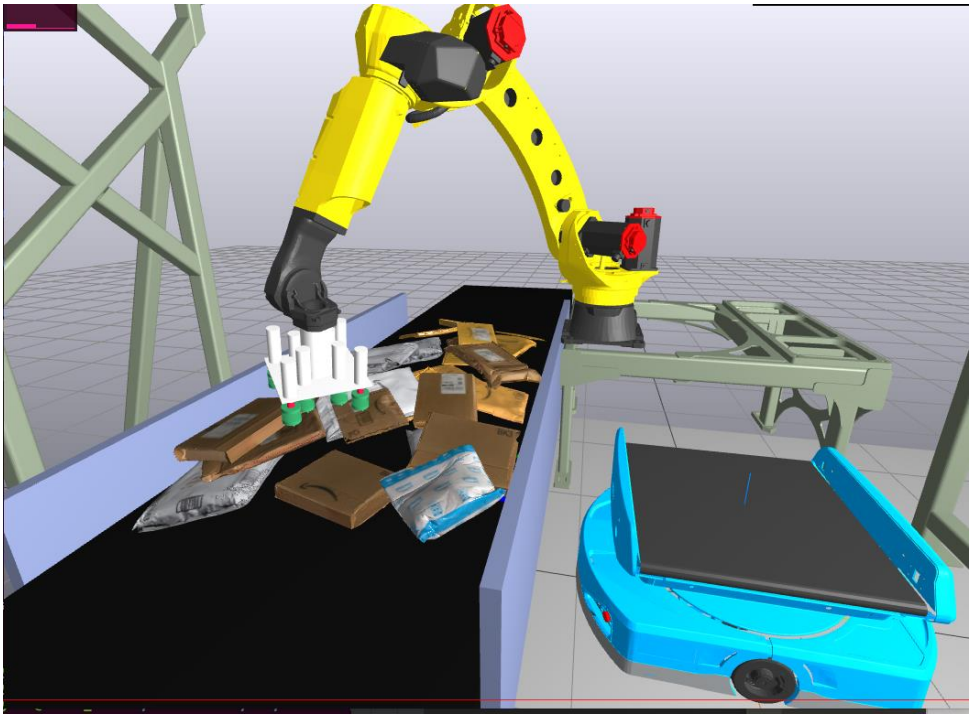

Рисунок 1.9 – Комп’ютерний прототип робота

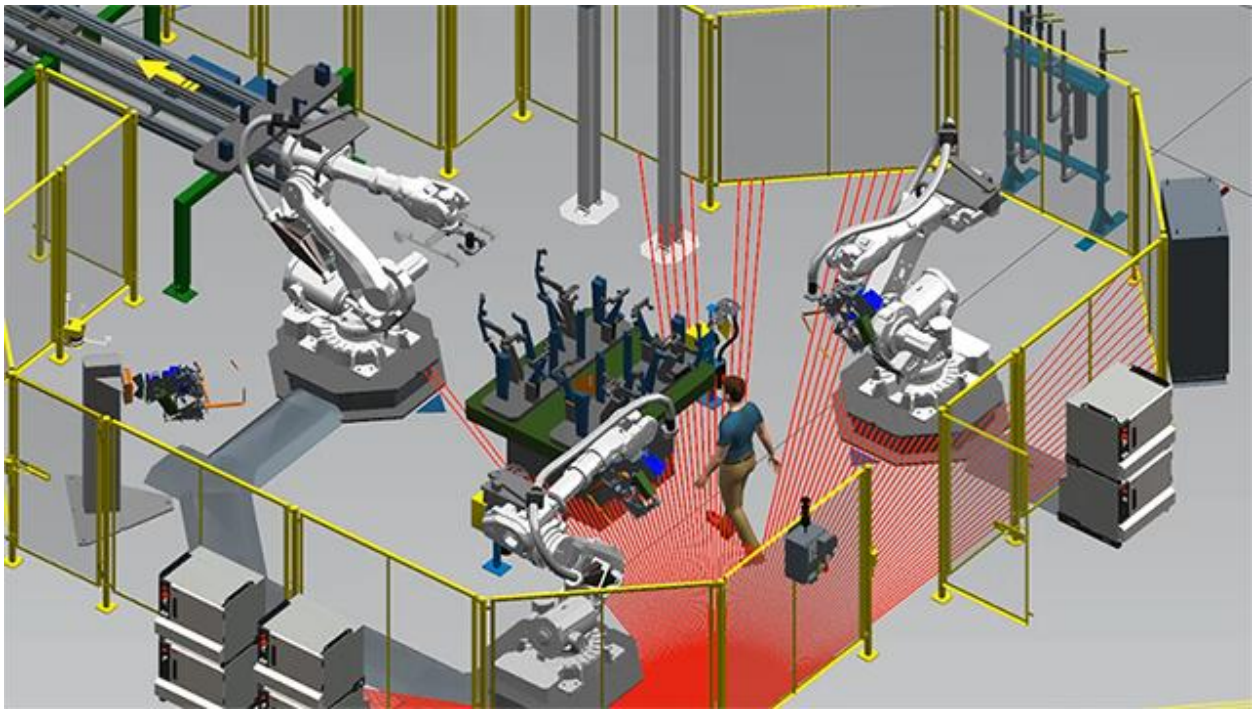

Рисунок 1.10 – Безпека людини під час симуляції

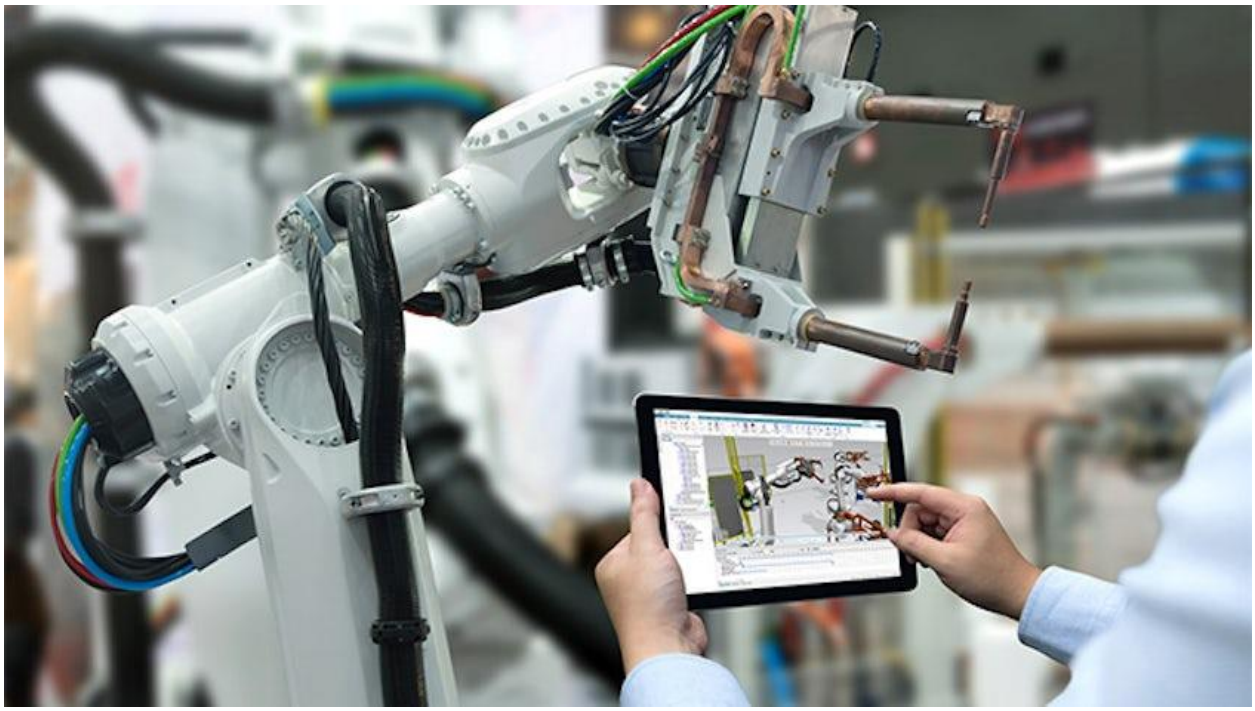

Рисунок 1.11 – Програмування робота на його комп’ютерній моделі

– розширене навчання: забезпечує безпечне та контрольоване середовище для навчання операторів і розробників – використання моделювання віртуальної реальності для навчання працівників взаємодії з роботами (рис 1.12) [18];

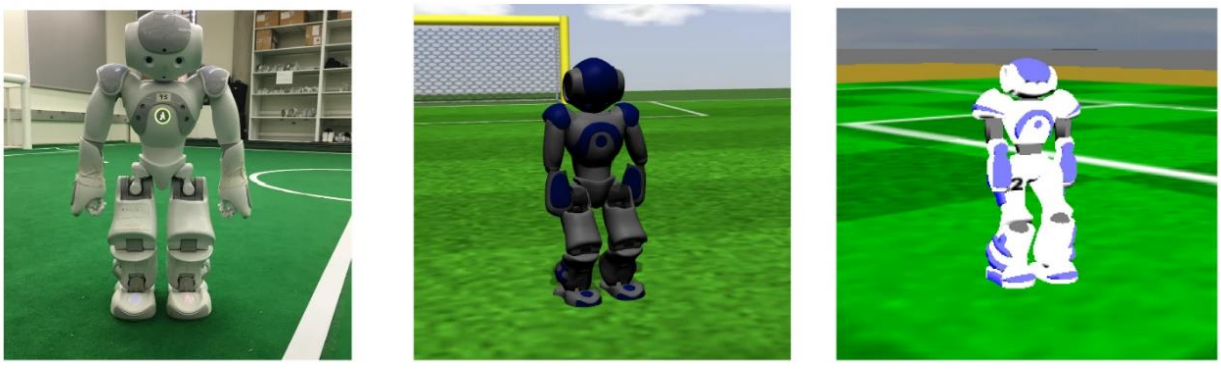

Рисунок 1.12 – Навчання комп’ютерної моделі робота

– оптимізація: допомагає оптимізувати роботизовані системи для кращої продуктивності та ефективності, а саме моделювання різних робочих умов для пошуку оптимальних налаштувань роботизованої руки (рис. 1.13) [19].

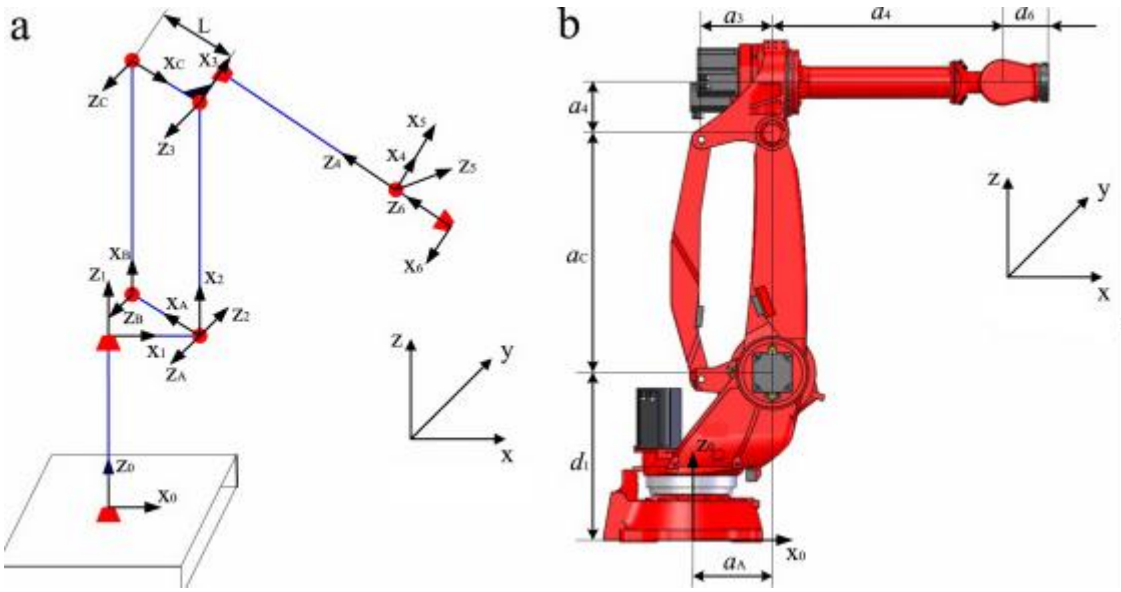

Рисунок 1.13 – Оптимізація роботи робота

Недоліками комп’ютерного моделювання в робототехніці є:

– масштабованість і складність: моделювання масштабування для складних систем без зниження продуктивності може бути складним, наприклад, моделювання цілої фабрики з декількома взаємодіючими роботами може потребувати інтенсивних обчислень;

– реалізм і точність: симуляції не завжди можуть ідеально відтворювати умови реального світу, що призводить до розбіжностей – робот, який добре працює в симуляції, може зіткнутися з непередбаченими проблемами в реальному світі;

– початкова вартість налаштування: високі початкові витрати на програмне забезпечення, апаратне забезпечення та кваліфікований персонал для налаштування та підтримки моделювання, наприклад, інвестиції у високопродуктивні обчислювальні ресурси та спеціалізоване програмне забезпечення для моделювання;

– вимоги до даних: точне моделювання потребує великої кількості даних, які може бути важко отримати, наприклад, збір детальних даних про навколишнє середовище для створення реалістичних імітаційних моделей;

– надмірна залежність від симуляції: може призвести до нехтування реальним тестуванням і перевіркою, приклад. припустимо, що робот буде ідеально працювати в реальному світі виключно на основі результатів моделювання [20].

1.5 Сучасні засоби для моделювання

Віртуальна реальність (VR) і доповнена реальність (AR) трансформують сферу робототехніки, покращуючи дизайн, навчання та експлуатаційні можливості [21].

Впровадження VR і AR у робототехніці має велике значення у майбутньому для наступних задач (рис. 1.14):

– дистанційне керування: VR і AR забезпечують захоплюючі інтерфейси для дистанційного керування роботами, наприклад, оператори можуть використовувати гарнітури віртуальної реальності для керування роботами в небезпечних середовищах, таких як атомні станції чи місця катастроф, з безпечної відстані;

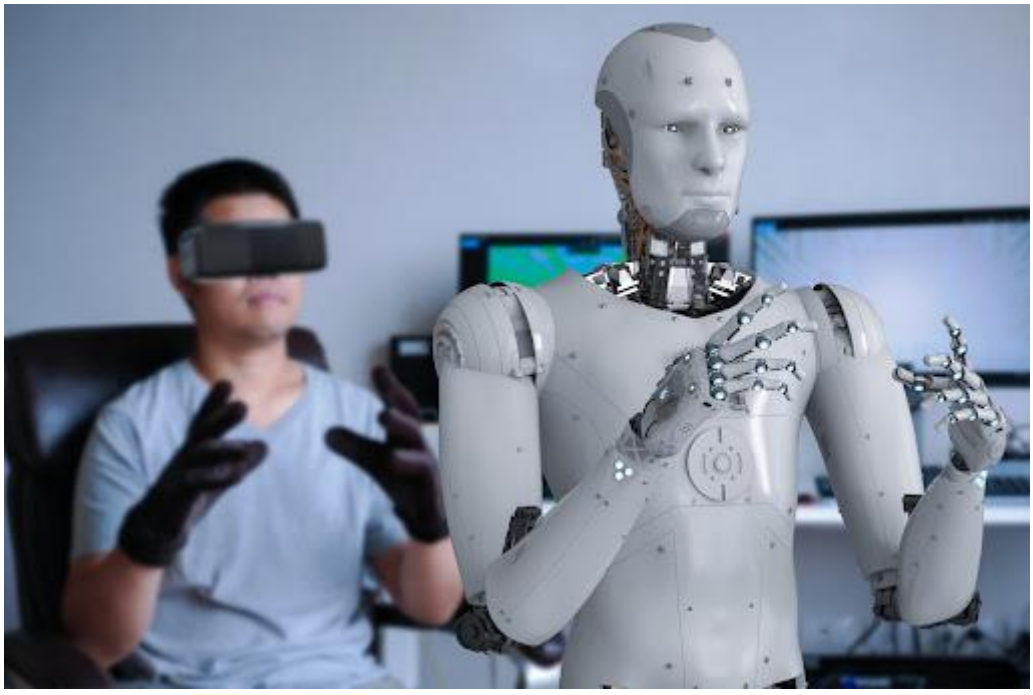

Рисунок 1.14 – Використання VR і AR для завдань робототехніки

– навчання та моделювання: VR і AR створюють реалістичні навчальні середовища для роботів-операторів і розробників – слухачі можуть практикувати керування роботами у віртуальному середовищі, отримуючи досвід без ризику пошкодити реальне обладнання;

– дизайн і прототипування: VR і AR дозволяють візуалізувати та маніпулювати роботами у віртуальному просторі, а саме, інженери можуть використовувати AR для накладення цифрових моделей на фізичні прототипи, що дозволяє коригувати та покращувати в реальному часі;

– технічне обслуговування та ремонт: AR надає покрокові візуальні інструкції щодо обслуговування та ремонту роботів – техніки можуть використовувати окуляри AR, щоб переглядати докладні посібники з ремонту, накладені на реального робота, що покращує точність і ефективність;

– покращена взаємодія людини і робота: AR може покращити взаємодію між людьми та роботами, надаючи контекстну інформацію, наприклад, працівники можуть використовувати AR, щоб отримувати дані та інструкції в реальному часі від роботів, сприяючи кращій співпраці над завданнями.

Перевагами у впровадженні VR, AR для завдань робототехніки є:

– покращена безпека: зменшує ризик нещасних випадків, дозволяючи дистанційне керування та надаючи докладні посібники з обслуговування;

– економія коштів: зниження витрат на навчання та створення прототипів завдяки використанню віртуальних середовищ замість фізичних налаштувань;

– підвищена ефективність, надаючи дані та інструкції в реальному часі, а також оптимізуючи процеси проектування;

– розширене навчання: забезпечує захоплюючий та інтерактивний досвід навчання, що сприяє кращому утриманню та розвитку навичок.

1.6 Висновки до розділу 1

В ході написання 1 розділу було проведено огляд сучасних джерел за напрямком Індустрії 5.0, а також комп’ютерного моделювання і було вирішено наступні завдання:

– Індустрія 5.0 та роль моделювання у ній – виділено основні аспекти сучасного розвитку промисловості у епоху Ідустрії 5.0, а також роль моделювання в цю епоху;

– симуляція та моделювання у епоху Індустрії 5.0 – розглянуто види моделювання у сучасному світі і їх основні напрямки;

– комп’ютерне моделювання та його роль у Індустрії 5.0 – окремо можна виділити саме цей напрямок, який має ряд своїх завдань для розвитку сучасної робототехніки за рахунок побудови більш дешевих комп’ютерних моделей, ніж реальні прототипи;

– переваги та недоліки комп’ютерного моделювання у завданнях робототехніки – виділено основні плюси та мінуси використання моделювання в робототехніці;

– виділено сучасні засоби для моделювання, такі як VR та AR, що дозволять більш докладно здійснювати завдання моделювання, керування та слідкування за робототехнічними системами.

2 ОБҐРУНТУВАННЯ ОСНОВНИХ ЕЛЕМЕНТІВ СИСТЕМИ МОДЕЛЮВАННЯ

2.1 Вибір середовища моделювання

Середовище моделювання роботів WeBots [22] має свої унікальні переваги та недоліки порівняно з іншими популярними середовищами моделювання, такими як Gazebo [23], V-REP (тепер CoppeliaSim) [24], та MATLAB/Simulink [25].

Переваги симулятора WeBots [26]:

– відкритий код: WeBots є програмним забезпеченням з відкритим кодом, що дозволяє користувачам модифікувати та розширювати його функціональність;

– підтримка різних мов програмування: підтримує C, C++, Python, Java, MATLAB та ROS2;

– інтерактивність: користувачі можуть взаємодіяти з моделлю під час моделювання;

– широкий спектр роботів та сенсорів: включає багато вбудованих моделей роботів та сенсорів;

– експорт у різні формати: можливість експортувати моделі у VRML, X3D, HTML-сцени та відео.

Недоліки:

– обмежена підтримка великих симуляцій: може бути менш ефективним для дуже великих або складних симуляцій порівняно з іншими середовищами;

– менша спільнота користувачів: порівняно з Gazebo, спільнота користувачів WeBots менша, що може ускладнити пошук допомоги.

Переваги Gazebo [27]:

– широка підтримка: велика спільнота користувачів та активна підтримка розробників;

– інтеграція з ROS: тісна інтеграція з ROS, що робить його ідеальним для розробки робототехнічних систем;

– масштабованість: підходить для великих та складних симуляцій.

Недоліки:

– складність налаштування: може бути складним у налаштуванні та використанні для новачків;

– вимогливість до ресурсів: вимагає значних обчислювальних ресурсів.

Переваги V-REP (CoppeliaSim) [28]:

– гнучкість: підтримує різні мови програмування та має потужний API4;

– модульність: можливість створення складних симуляцій з використанням модулів;

– реалістична фізика: високий рівень реалістичності фізичних симуляцій.

Недоліки:

– крута крива навчання: може бути складним для освоєння новачками;

– ліцензійні обмеження: деякі функції доступні лише у платній версії.

Переваги MATLAB/Simulink [29]:

– потужні інструменти для аналізу: включає потужні інструменти для аналізу та візуалізації даних;

– інтеграція з іншими інструментами MATLAB: легка інтеграція з іншими інструментами MATLAB;

– підтримка різних типів роботів: підтримує моделювання різних типів роботів.

Недоліки:

– вартість: висока вартість ліцензії;

– складність: може бути складним для новачків.

Провівши аналіз, можна зробити висновок що WeBots є відмінним вибором для тих, хто шукає потужне та гнучке середовище моделювання з відкритим кодом, яке підтримує різні мови програмування та має широкий

спектр вбудованих моделей роботів та сенсорів. Це середовище особливо підходить для навчальних цілей та швидкого прототипування.

Основні можливості WeBots:

– реалістична фізика: WeBots використовує фізичні рушії, такі як ODE (Open Dynamics Engine), для забезпечення реалістичної симуляції руху та взаємодії роботів з навколишнім середовищем;

– підтримка різних роботів: включає вбудовані моделі різних типів роботів, таких як мобільні роботи, маніпулятори, дрони та навіть біонічні роботи;

– сенсори та актуатори: підтримує широкий спектр сенсорів (камери, лідари, GPS, IMU) та актуаторів (моторів, сервоприводів), що дозволяє створювати складні симуляції;

– інтеграція з ROS: WeBots має вбудовану підтримку ROS (Robot Operating System), що дозволяє легко інтегрувати симуляції з реальними робототехнічними системами;

– мультиплатформність: працює на Windows, macOS та Linux, що робить його доступним для широкого кола користувачів.

Використання WeBots у навчанні та дослідженнях полягає в тому, що WeBots широко використовується в академічних та дослідницьких установах для навчання студентів основам робототехніки та для проведення досліджень у цій галузі. Завдяки своїй гнучкості та потужності, WeBots дозволяє студентам та дослідникам швидко створювати та тестувати нові алгоритми та концепції.

Приклади використання WeBots:

– навчальні проекти: студенти можуть використовувати WeBots для створення симуляцій простих роботів, таких як лінійні роботи або роботи для змагань;

– дослідницькі проекти: дослідники можуть використовувати WeBots для моделювання складних робототехнічних систем, таких як автономні транспортні засоби або роботи для дослідження космосу;

– індустріальні застосування: WeBots також використовується в індустрії для моделювання та тестування робототехнічних систем перед їх впровадженням у виробництво.

2.2 Створення простого світу у WeBots з використанням робота

Проведемо розробку першої моделі для симуляції. Натиснемо пункт меню File – New Project Directory – Next (рис. 2.1). Далі потрібно обрати теку, куди буде збережено проєкт, ввести його ім’я.

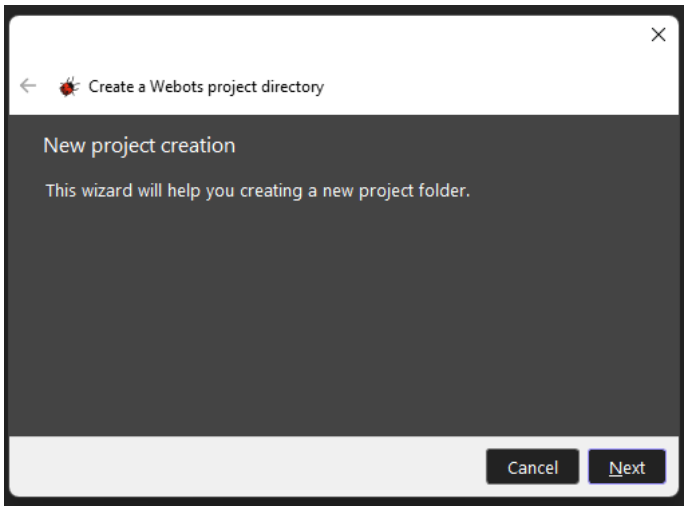

Рисунок 2.1 – Вікно створення проєкту

Далі ввести ім’я світу, обрати створення квадратної арени (рис. 2.2).

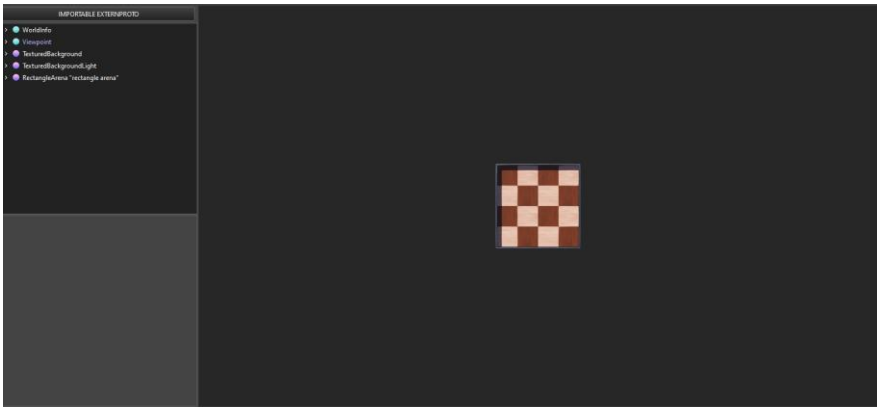

Рисунок 2.2 – Новий World

Кожен з об’єктів сцени називається вузол (Node).

Створимо додатковий об’єкт (Node). Для цього потрібно обрати вузол RectangleArena та натснути клавішу AddObject або правою кнопкою миші по ньому та Add New.

Далі створимо власний об’єкту типу Solid, тобто твердий об’єкт, що має певні властивості (рис. 2.3).

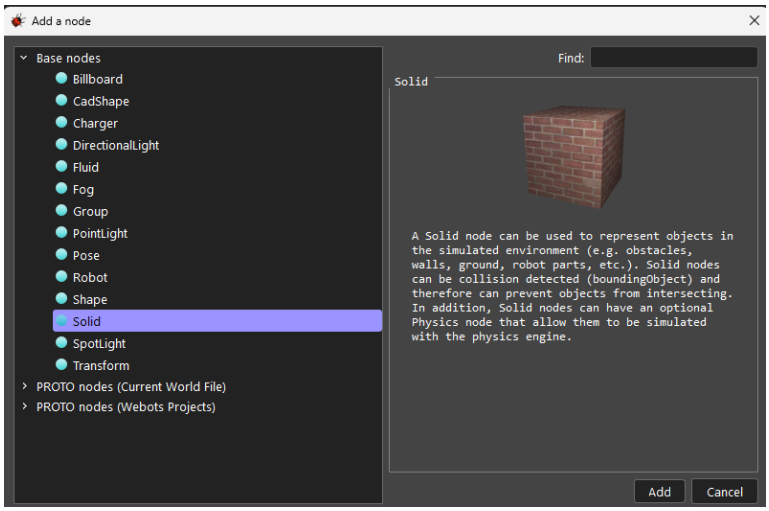

Рисунок 2.3 – Створення об’єкту типу Solid

У режимі симуляції (Simulation View) ліворуч, потрібно обрати параметр children (рис. 2.4) та натиснемо кнопку Add new object.

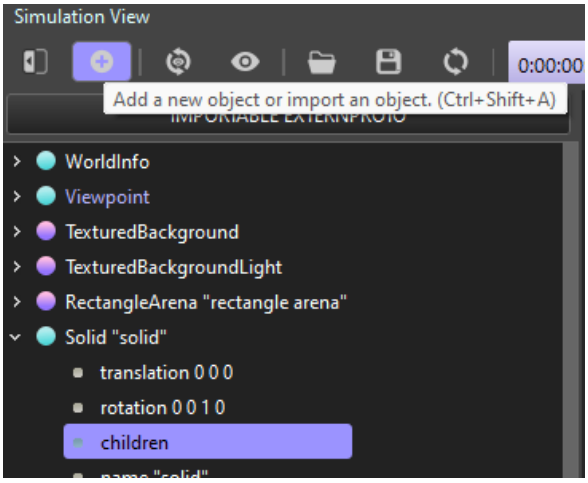

Рисунок 2.4 – Налаштування об’єкту типу Solid

Потрібно додати ще один дочірній вузол під назвою Shape (рис. 2.5).

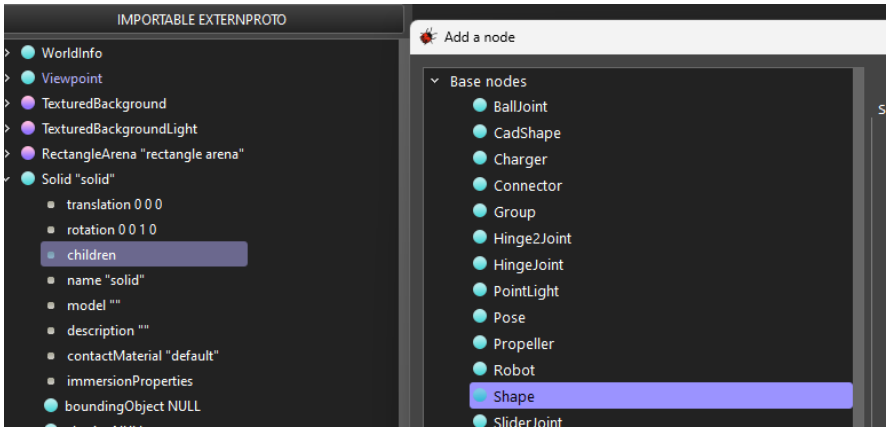

Рисунок 2.5 – Додавання Node під назвою Shape

В налаштуваннях цього вузлу оберемо геометрію geometry (рис. 2.6) та натиснемо по ньому 2 рази лівою кнопкою миші.

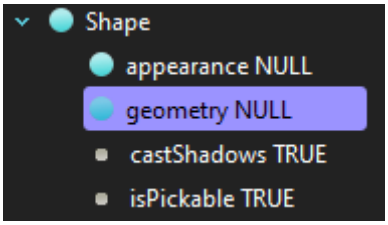

Рисунок 2.6 – Обирання геометрії

Далі можна обрати будь-яку форму, наприклад сферу (Sphere). Всі об’єкти додаються у координати (0;0;0).

Також є можливість змінити параметри цієї сфери (рис. 2.7), де перший параметр відповідає за радіус сфери, інші два – за згладжування.

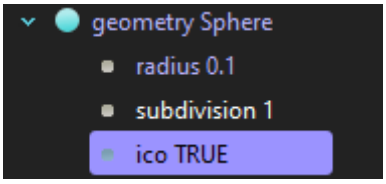

Рисунок 2.7 – Налаштування сфери

Далі змінимо зовнішній вигляд (appearance) – обрати PBRAppearance – візуальні властивості геометрії.

В середині цього вузла також є багато налаштувань (рис. 2.8). Оберемо параметр roughness 1, а також metalness 0, а також колір червоний (1; 0; 0) (рис. 2.9).

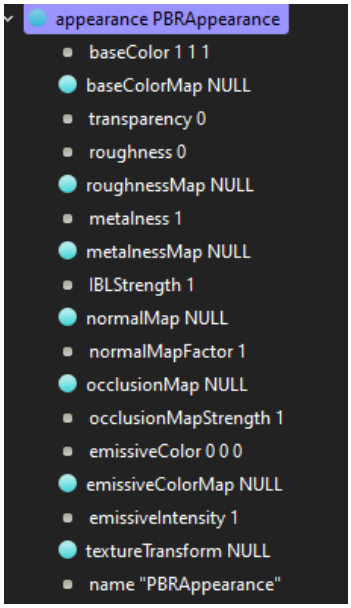

Рисунок 2.8 – Налаштування PBRAppearance

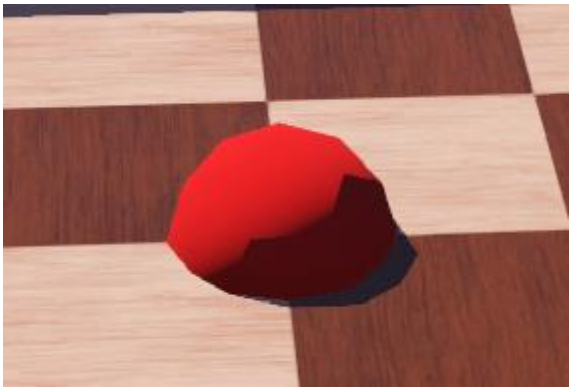

Рисунок 2.9 – Доданий об’єкт

Далі додаємо параметр обмежуючого об’єкту (boundingObject) (рис. 2.10).

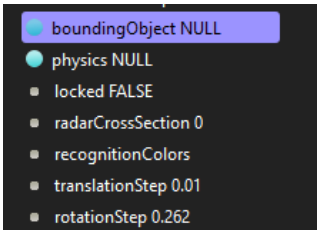

Рисунок 2.10 – Обмежуючий об’єкт

Щоб обрати створену сферу у якості такого об’єкту, потрібно в параметрах сфери визначити його, тобто в параметр DEF ввести певне ім’я, наприклад Ball (рис. 2.11).

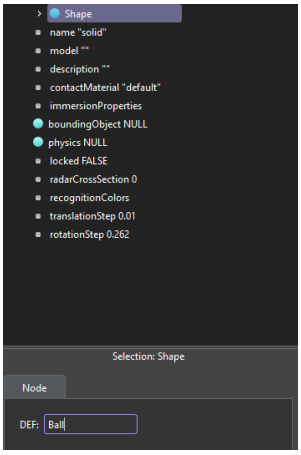

Рисунок 2.11 – Визначення об’єкту Ball

Далі натиснути 2 рази по boundingObject та з меню Use обрати щойно створений об’єкт (рис. 2.12).

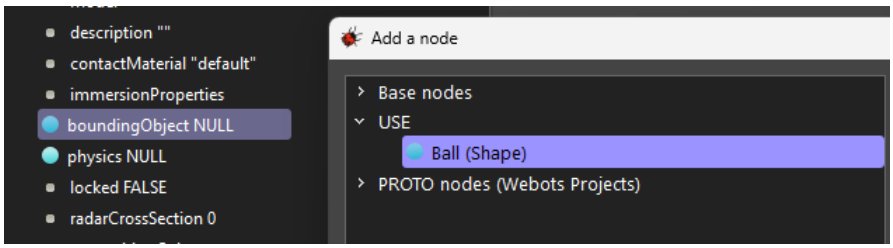

Рисунок 2.12– Обирання створеного об’єкту

Далі додати цьому об’єкту фізику (physics), наприклад, гравітацію (рис. 2.13).

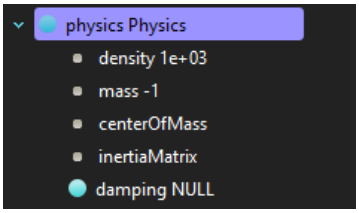

Рисунок 2.13 – Параметри фізики для об’єкту

При проведенні симуляції (клавіша Run Simulation) (рис. 2.14), сфера буде падати на землю, маючи фізику (гравітація) та інерцію. Для того, щоб скинути симуляцію використати клавішу Reset Simulation.

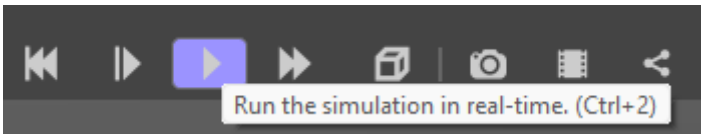

Рисунок 2.14 – Запуск симуляції

Додамо робота з існуючих. Для цього натиснути клавішу Add a New Object та обрати з існуючих об’єктів (рис. 2.15) – PROTO nodes (Webots Projects), наприклад, мобільний робот Robotino (рис. 2.16).

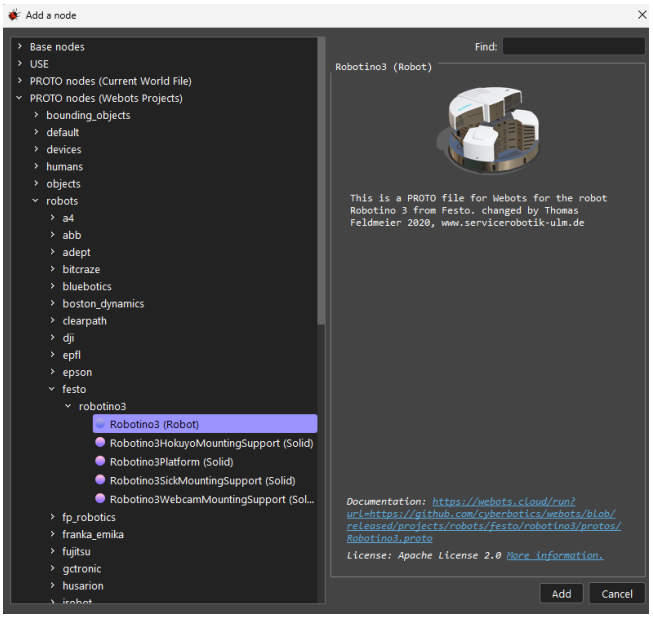

Рисунок 2.15 – Додавання існуючих об’єктів

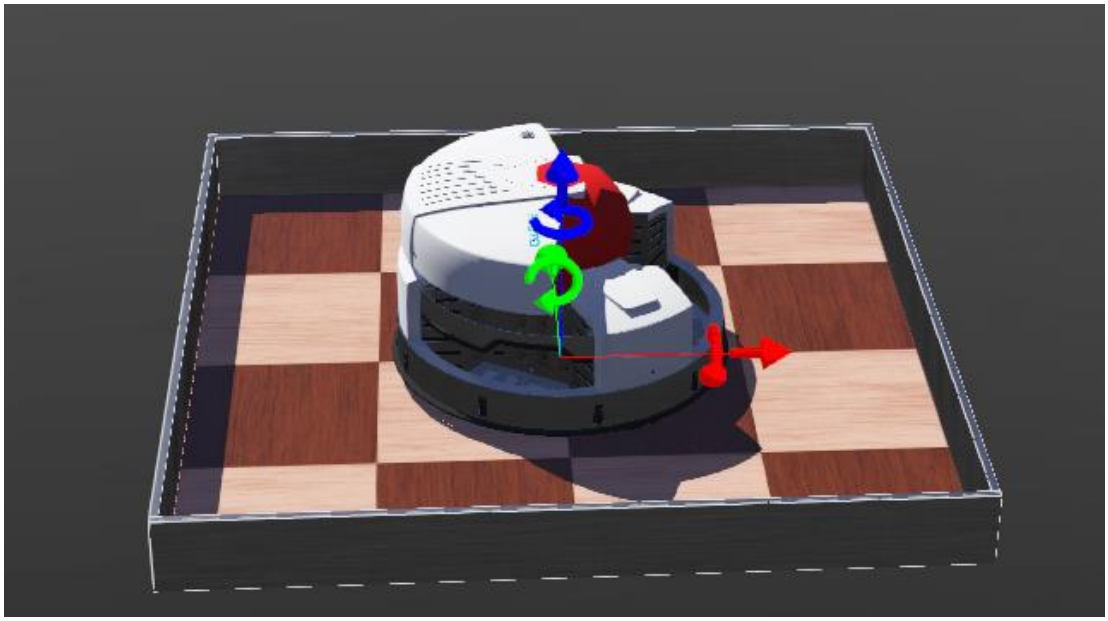

Рисунок 2.16 – Додавання робота Robotino

Далі для зручності змінити розміри сцени на 3 м × 3 м (рис. 2.17).

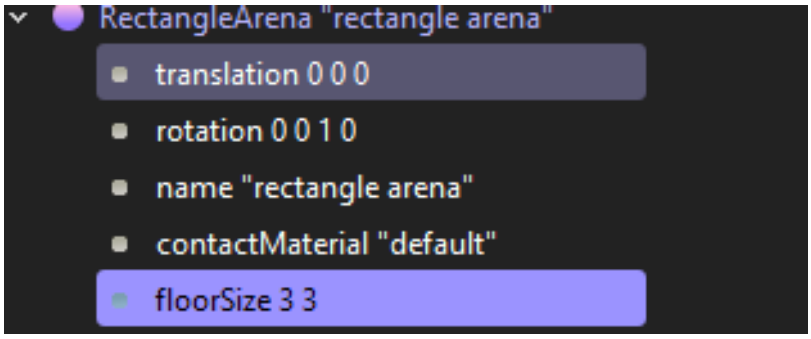

Рисунок 2.17 – Зміна розміру сцени на 3 м × 3 м

Після запуску симуляції, робот починає свій рух, використовуючи свої інфрачервоні датчики відстані. Робот також бачить додані об’єкти, тому що вони мають фізичні властивості. У випадку, якщо робот не бачить датчиками об’єкт, він може його штовхнути.

Зараз робот використовує вже розроблений контролер (controller). Щоб його подивитися, потрібно обрати відповідний параметр та натиснути клавішу Edit, після чого програма керування відкриється праворуч у редакторі тексту (Text Editor) (рис. 2.18).

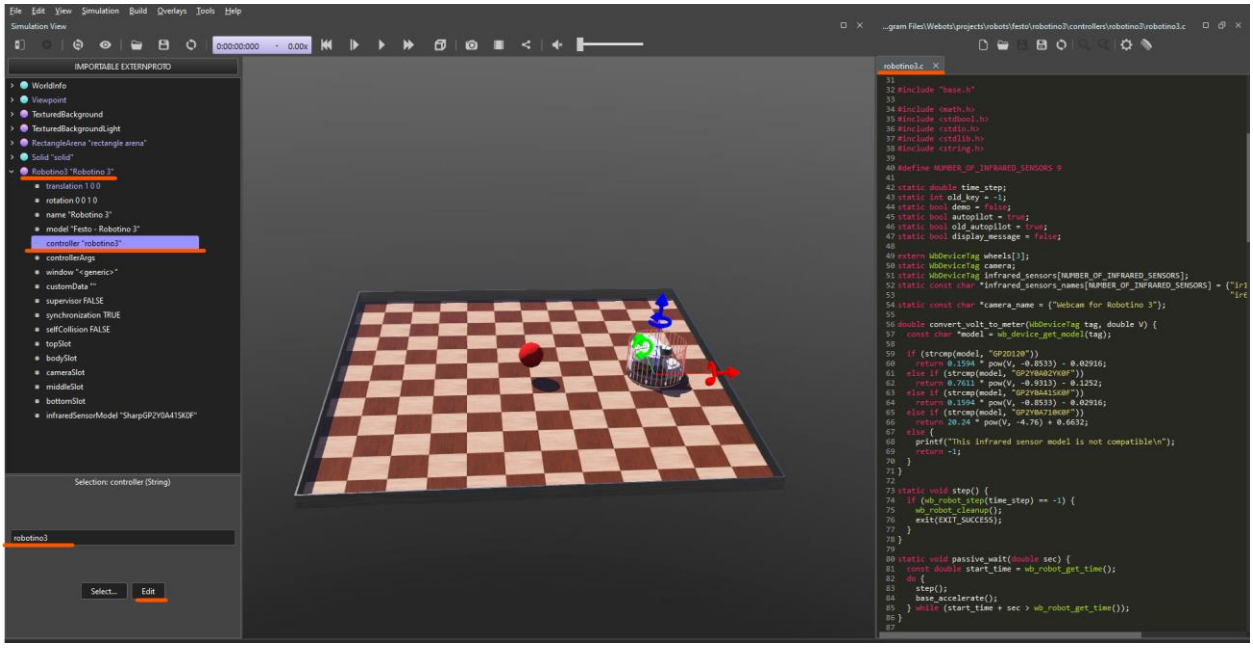

Рисунок 2.18 – Контролер обраного робота

Даний код написаний мовою C/C++, його можна замінити на код, написаний мовами Java та Python.

2.3 Теорія автоматичного управління

Знайдемо передавальну функцію постійного струму двигуна. Передавальна функція – це математичний опис відношення вихідного сигналу системи до вхідного сигналу [30].

Для двигуна, передавальна функція може бути виведена з основних рівнянь електромеханічної системи.

Електричне рівняння:

$$V(t) = L\frac{di(t)}{dt} + Ri(t) + e(t),$$

де $V(t)$ – прикладена напруга:

L – індуктивність обмотки;

R – опір обмотки;

i(t) – струм через обмотку;

e(t) –ЕРС (електрорушійна сила), яка пропорційна швидкості обертання двигуна.

Механічне рівняння:

$$J\frac{dw(t)}{dt} = k_t i(t) - bw(t),$$

де J – момент інерції ротора;

$w(t)$ – кутова швидкість ротора;

k_t– константа крутного моменту;

b – коефіцієнт в'язкого тертя.

Зв'язок між проти-ЕРС і швидкістю:

$$e(t) = k_b w(t),$$

де k_b – константа проти-ЕРС.

Якщо ми припустимо, що індуктивність L дуже мала і може бути знехтувана, то електричне рівняння спрощується до:

$$V(t) = Ri(t) + k_b w(t)\ .$$

Тепер ми можемо використати це рівняння разом з механічним рівнянням для виведення передавальної функції двигуна як відношення кутової швидкості $w(s)$ до прикладеної напруги $V(s)$, де s – оператор Лапласа.

Припустимо, що струм $i(t)$ і швидкість $w(t)$ є вихідними сигналами, а $V(t)$ – вхідним сигналом.

Перетворимо рівняння до форми Лапласа:

$$V(s) = RI(s) + sLI(s) + k_b\Omega(s),$$

$$sJ\Omega(s) = k_t I(s) - b\Omega(s),$$

де $I(s)$ та $\Omega(s)$ – перетворення Лапласа для струму і швидкості відповідно.

Якщо ми знехтуємо індуктивністю (L), то перше рівняння спрощується до:

$$V(s) = RI(s) + k_b\Omega(s).$$

Тепер ми можемо вирішити це рівняння для $I(s)$:

$$I(s) = \frac{V(s)}{R+k_b s}.$$

Підставляючи це у друге рівняння, отримуємо:

$$sJ\Omega(s) = k_t \frac{V(s)}{R+k_b s} - b\Omega(s)\ .$$

Розв'язуючи це рівняння для $\Omega(s)$, отримуємо передавальну функцію двигуна:

$$G(s) = \frac{\Omega(s)}{V(s)} = \frac{k_t}{s(Js+b)(R+k_b s)}.$$

Ця передавальна функція показує взаємозв'язок між прикладеною напругою і кутовою швидкістю двигуна. Вона дозволяє аналізувати динамічну поведінку системи та проектувати контролер для досягнення бажаного відгуку системи.

Нехай параметри нашого двигуна будуть наступними:

– опір обмотки R = 2 Ом;

– константа крутного моменту k_t = 0,01 Н·м;

– константа проти-ЕРС k_b = 0,01 В·с/рад;

– момент інерції ротора J = 0,01 кг·м²;

– коефіцієнт в’язкого тертя b = 0,1 Н·с/м^2.

Тепер ми можемо підставити ці значення у передавальну функцію:

$$G(s) = \frac{0{,}01}{s(0{,}01s+0.1)(2+0{,}01s)} \ ,$$

$$G(s) = \frac{0{,}01}{0{,}02s^2+0{,}201s+0{,}2}.$$

Отримана передавальна функція показує, як кутова швидкість двигуна змінюється відносно прикладеної напруги.

Проведемо побудову перехідної функції системи (рис. 2.19).

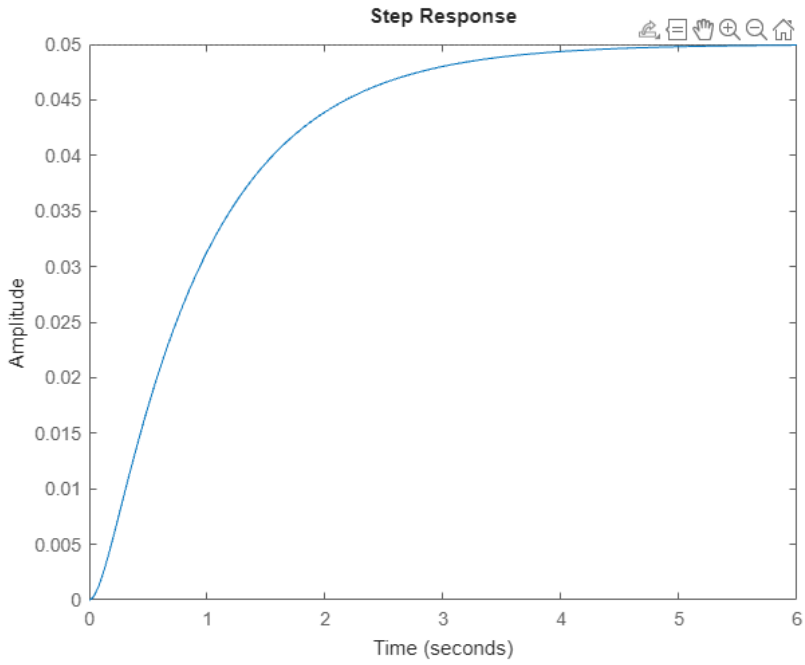

Рисунок 2.19 – Перехідна характеристика системи

Також перевіримо систему на стійкість за критеріями.

По-перше, за критерієм Гурвіца [31]. Для цього розробимо матрицю коефіцієнтів:

$$\Delta = \begin{bmatrix} 0{,}201 & 0 & 0 \\ 0{,}02 & 0{,}2 & 0 \\ 0 & 0{,}201 & 0 \end{bmatrix}.$$

Розрахуємо визначники:

$$\Delta_1 = 0{,}201,$$
$$\Delta_2 = 0{,}201 \cdot 0{,}2 - 0{,}02 \cdot 0 = 0{,}0402,$$
$$\Delta_3 = \Delta_2 \cdot 0 = 0.$$

Визначники матриці більші за 0, а також $a_0 > 0$, отже можна зробити висновок що система є стійкою за критерієм Гурвіца.

Проведемо дослідження стійкості за критерієм Михайлова:

$$G(s) = \frac{0{,}01}{0{,}02s^2 + 0{,}201s + 0{,}2},$$
$$G(jw) = \frac{0{,}01}{0{,}02(jw)^2 + 0{,}201jw + 0{,}2},$$
$$G(jw) = \frac{0{,}01}{-0{,}02w^2 + 0{,}201jw + 0{,}2} = \frac{0{,}01}{0{,}201jw - (0{,}02w^2 - 0{,}2)},$$
$$G(jw) = \frac{0{,}01}{0{,}201jw - (0{,}02w^2 - 0{,}2)} * \frac{0{,}201jw + (0{,}02w^2 - 0{,}2)}{0{,}201jw + (0{,}02w^2 - 0{,}2)},$$
$$G(jw) = \frac{0{,}01 * \left(0{,}201jw + (0{,}02w^2 - 0{,}2)\right)}{(0{,}201jw)^2 - (0{,}02w^2 - 0{,}2)^2} =$$
$$= \frac{0{,}00201jw + 0{,}0002w^2 - 0{,}002}{-0.0404w^2 - 0{,}0004w^4 + 0{,}08w^2 - 0{,}04},$$
$$G(jw) = \frac{0{,}00201jw + 0{,}0002w^2 - 0{,}002}{-0{,}0004w^4 + 0.0324w^2 - 0{,}04},$$
$$Re(jw) = \frac{0{,}0002w^2 - 0{,}002}{-0{,}0004w^4 + 0.0324w^2 - 0{,}04},$$
$$Im(jw) = \frac{0{,}00201w}{-0{,}0004w^4 + 0.0324w^2 - 0{,}04}.$$

Побудуємо графік кривої Михайлова [30] в середовищі інтерактивного розрахунку великих масивів числових даних. Для цього використаємо код нижче:

```
W=tf([0.01],[0.02 0.201 0.2])
syms s w
D= 0.02*s^2 + 0.201*s + 0.2;
s=j*w;
Djw=compose(D,s)
Re=[];Im=[];
for w=0:0.1:100
Djw=- w^2/50 + (w*201i)/1000 + 1/5;
Re1=real(Djw);
Im1=imag(Djw);
Re=[Re,Re1];
Im=[Im,Im1];
end
figure, plot (Re, Im)
grid on
```

Результат виконання побудови графіка представлено на рис. 2.20-2.21.

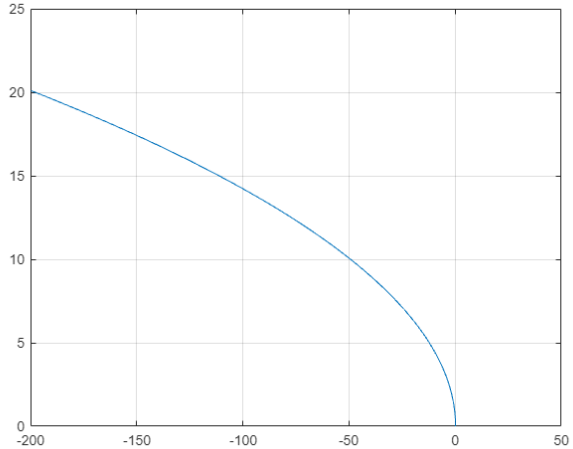

Рисунок 2.20 – Графік критерія Михайлова

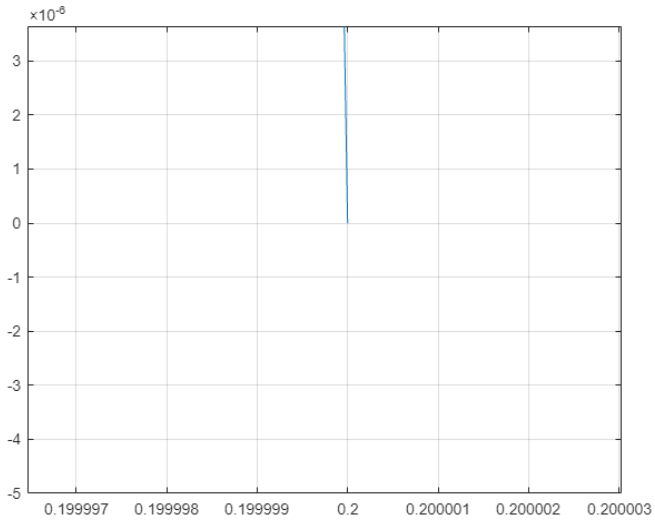

Рисунок 2.21 – Збільшений графік критерія Михайлова

Виходячи з графіку, можна зробити висновок що система є стійкою за критерієм Михайлова, так як крива послідовно обходить 2 квадранти, де 2 – ступінь полінома, а також не потрапляє у початок координат.

Також перевіримо систему за критерієм Найквіста [31].

Знайдемо полюси передавальної функції:

$$s_1 = -8{,}9302,$$
$$s_1 = -1{,}1198.$$

По-перше, всі корені є лівими, тобто система стійка за кореневим критерієм [30].

По-друге, для того, щоб система була стійкою за критерієм Найквіста, вона повинна охоплювати точку $(-1; j_0)$ l разів, де l – кількість правих коренів, яких в нас нуль [31]. Побудуємо графік (рис. 2.22).

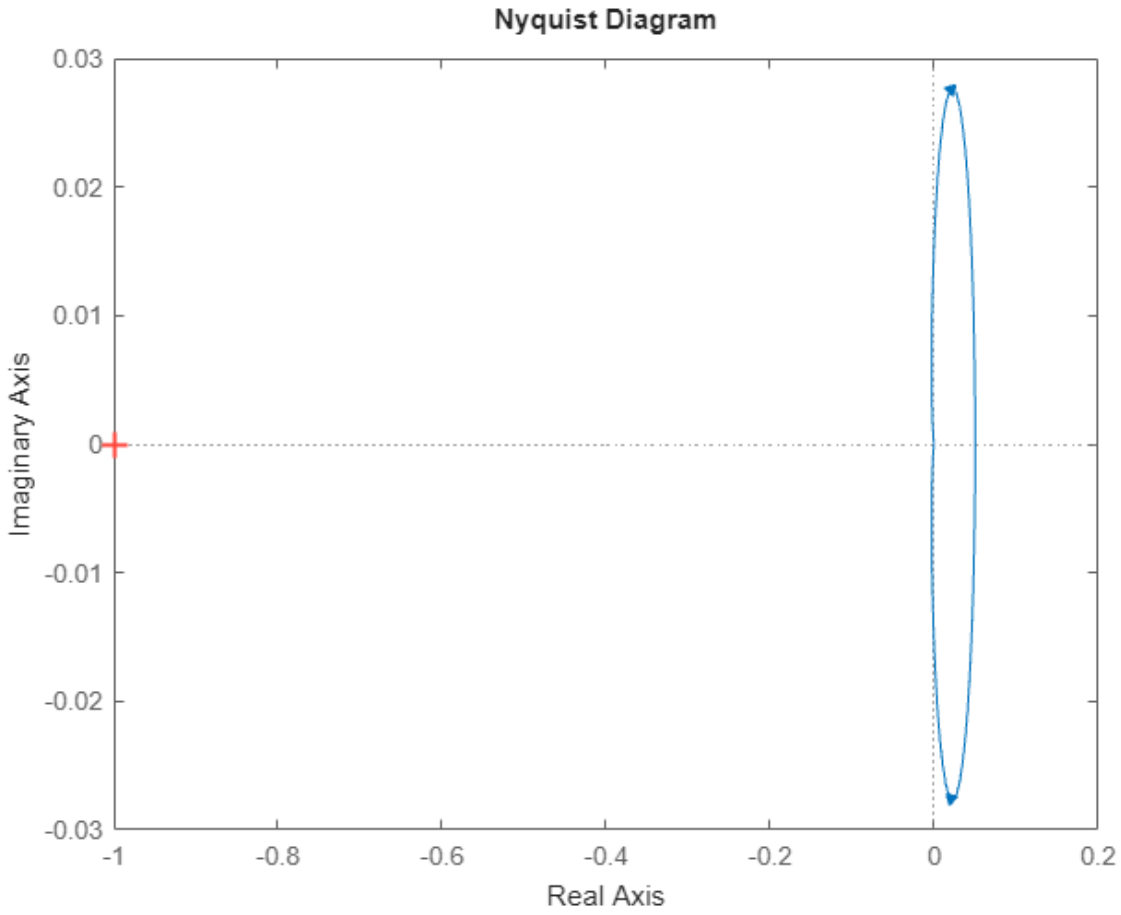

Рисунок 2.22 – Графік критерія Найквіста

Як видно з графіку, годограф охоплює точку 0 разів і це означає що система є стійкою за критерієм Найквіста [31].

2.4 Висновки до розділу 2

В результаті виконання 2 розділу було проведено порівняння різних середовищ моделювання роботи роботів, таких як WeBots, Gazebo, CoppeliaSim та MATLAB/Simulink. Кожна з цих систем має свої переваги та недоліки.

В якості середовища моделювання було обрано саме WeBots завдяки можливості провести в ньому моделювання власної моделі робота та коректного моделювання датчиків.

Також було розглянуто створення простого світу для моделювання робота, а також додавання інших елементів оточення, додаючи цим елементам фізичних властивостей, таких як силу тяжіння та масу.

Далі було розглянуто питання теорії автоматичного управління та знайдено передавальну функцію двигуна.

На основі отриманої передавальною функції було перевірено її стійкість за алгебраїчними та частотними критеріями

3 РОЗРОБЛЕННЯ МОДЕЛІ МОБІЛЬНОГО РОБОТА ТА ЙОГО СЕНСОРНОЇ СИСТЕМИ У СЕРЕДОВИЩІ WEBOTS

3.1 Створення світу для моделювання роботи сенсорної системи мобільного робота

Створимо новий проект (рис. 3.1) для симуляції, а саме нову теку для проекту, а також оберемо світ з додаванням квадратної арени.

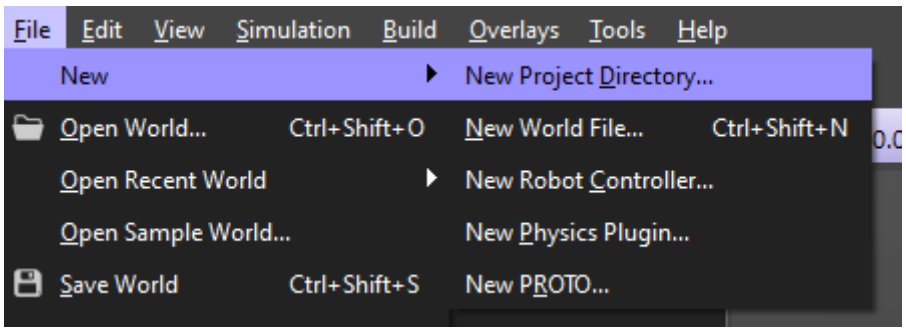

Рисунок 3.1 – Створення нового проекту

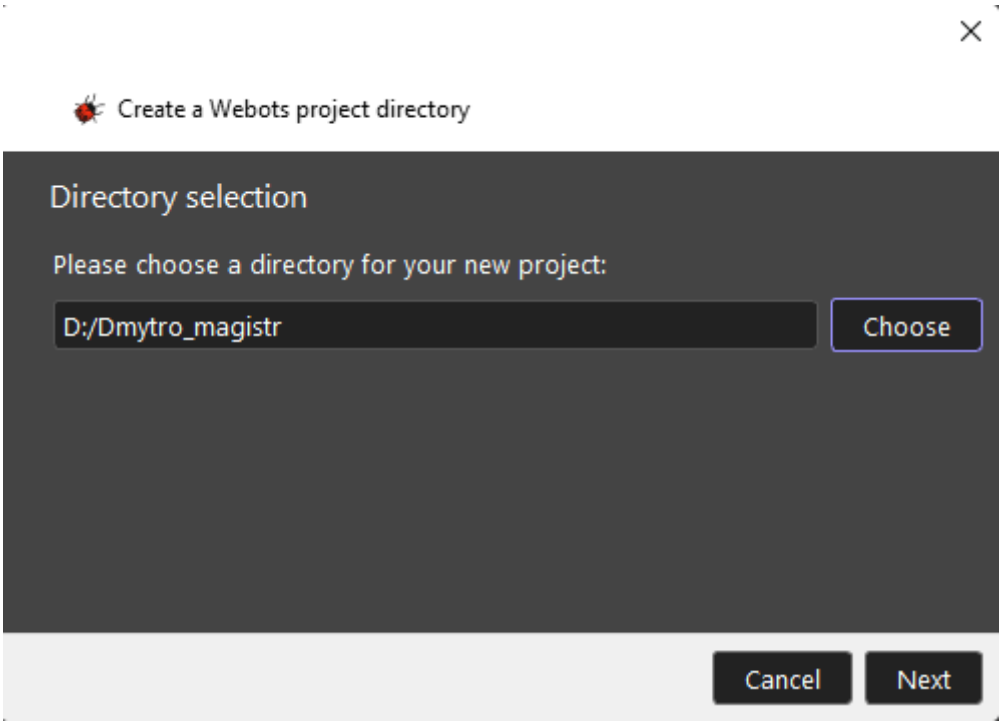

Рисунок 3.2 – Створення теки для проекту

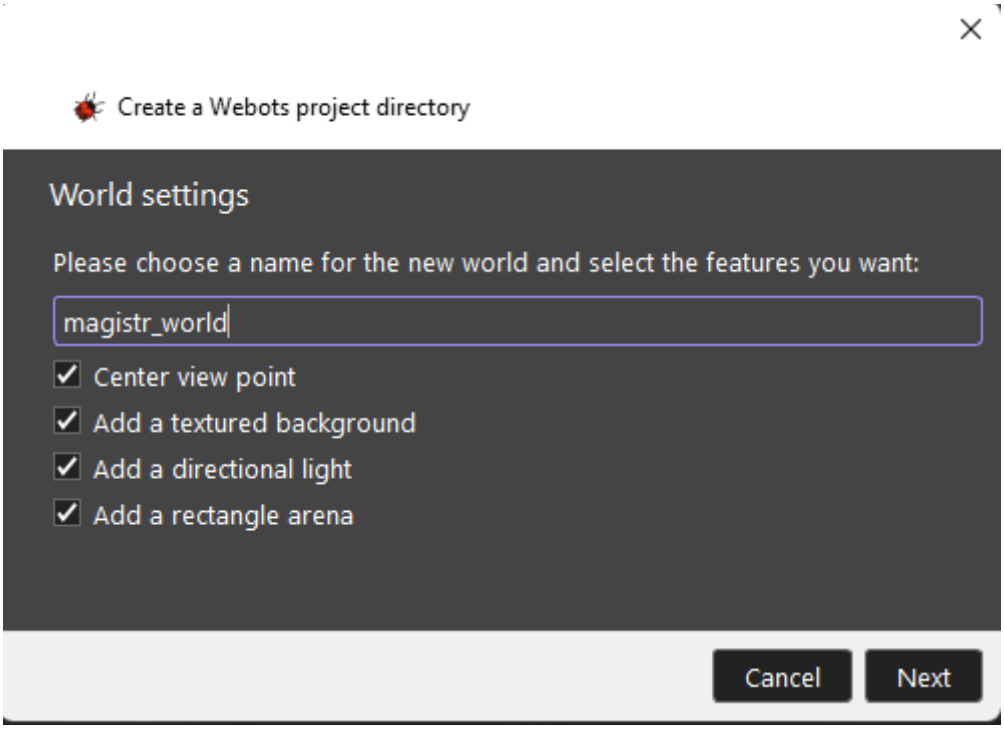

Рисунок 3.3 – Створення світу з квадратною ареною

Результат створення світу показано на рис. 3.4.

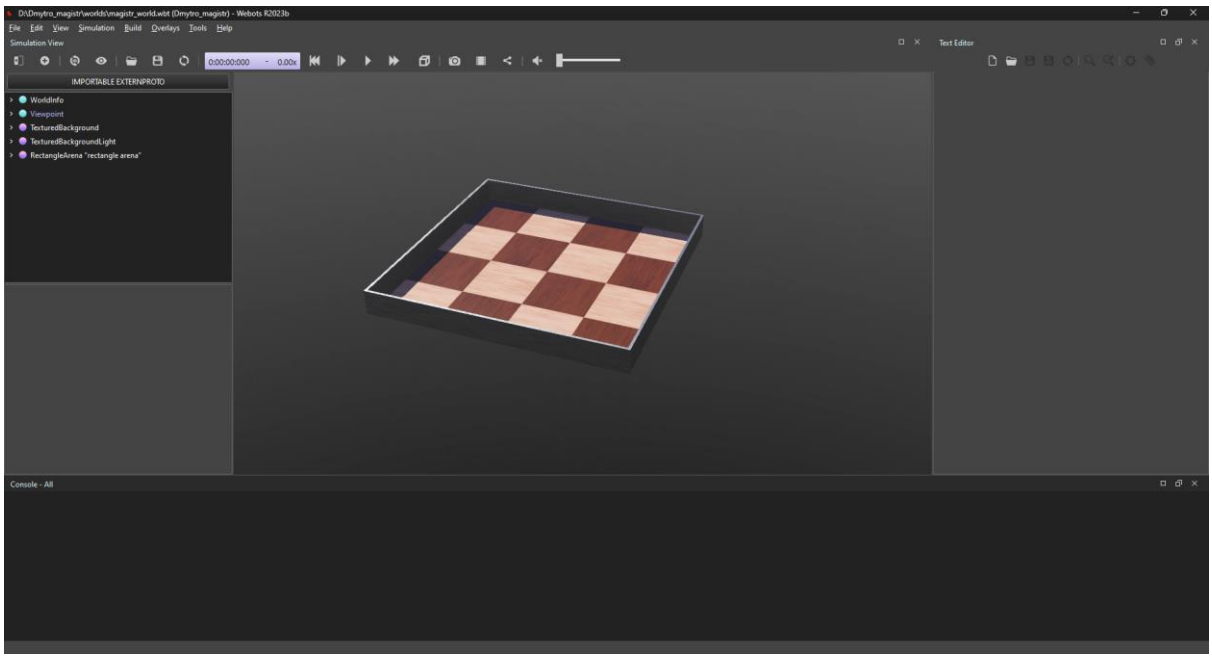

Рисунок 3.4 – Створений світ для моделювання робота

Змінимо його масштаб у Simulation View з 1 м × 1 м на 10 м × 10 м (рис. 3.5).

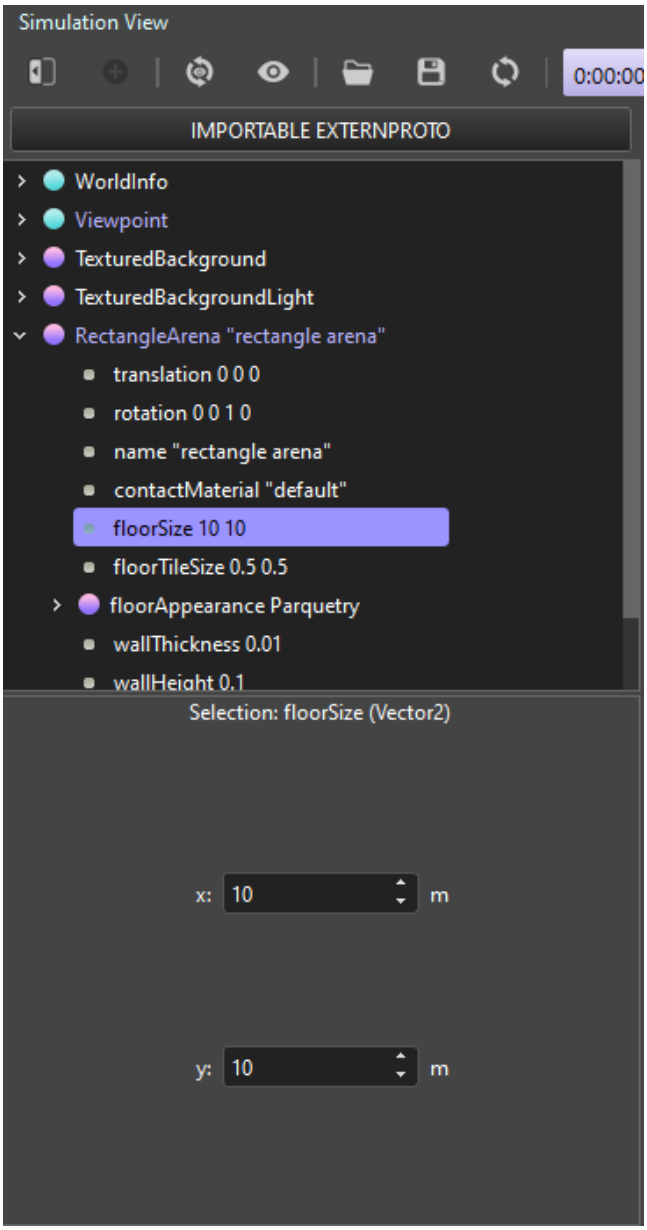

Рисунок 3.5 – Зміна масштабу арени

3.2 Моделювання мобільного робота

3.2.1 Створення робота

По-перше, потрібно додати вузол робота (рис. 3.6). Для цього потрібно обрати Base Nodes – Robot.

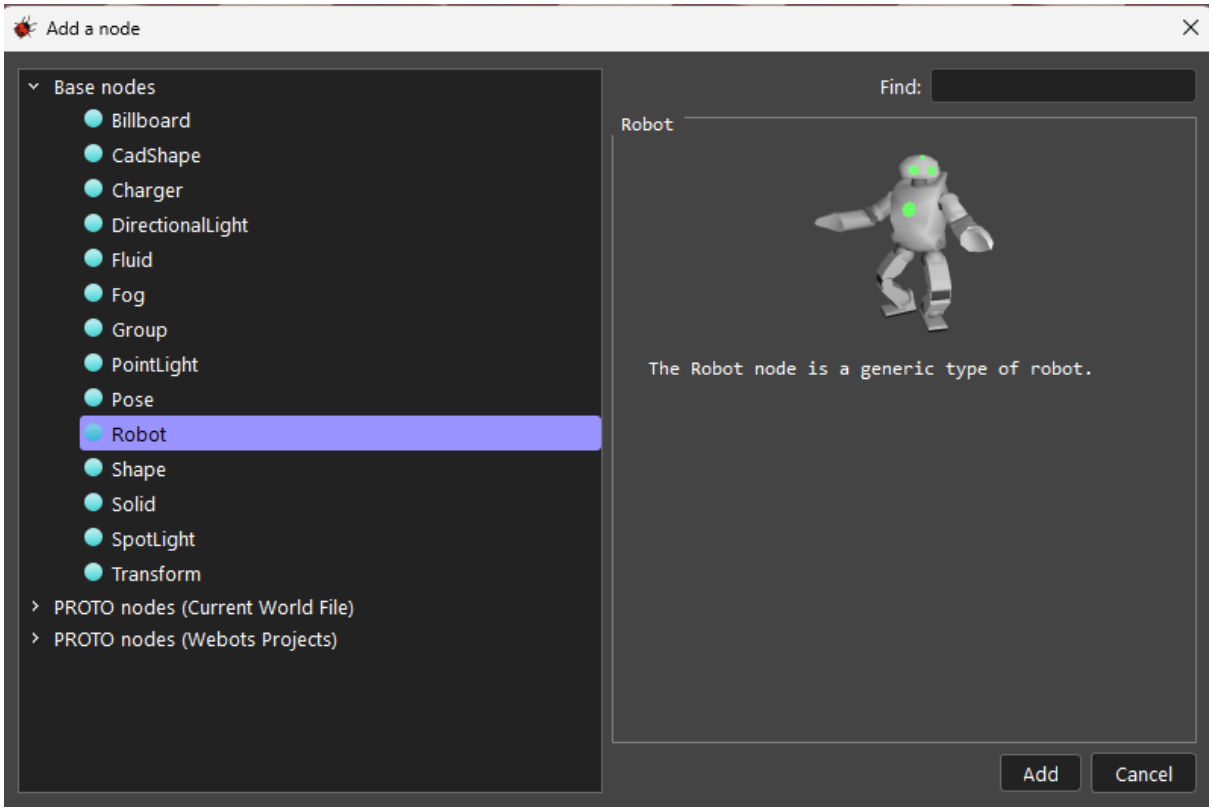

Рисунок 3.6 – Додавання вузлу для створення робота

Далі додамо у параметр children базовий вузол Transform (рис. 3.7) для групування елементів та визначання системи координат для дочірніх елементів (кожного з елементів робота).

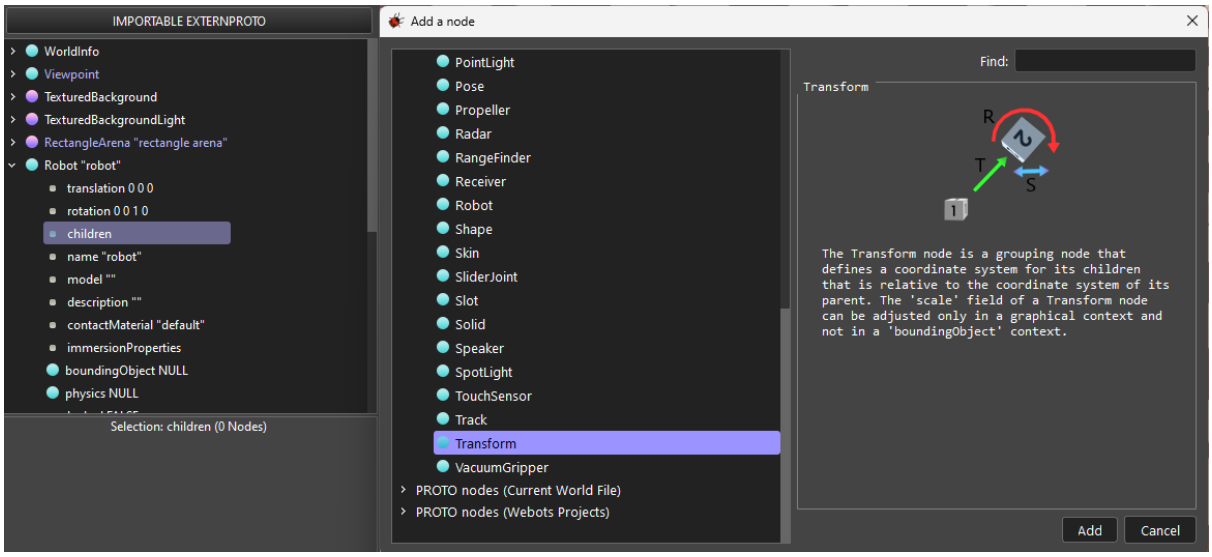

Рисунок 3.7 – Додавання вузлу трансформації

Далі потрібно створити тіло робота, для цього додати дочірній вузол Shape та обрати параметри appearance – PBRAppearance (рис. 3.8) для визначення візуальних властивостей геометрії, а також geometry – Box для створення прямокутного паралелепіпеда, що й буде тілом робота (рис. 3.9).

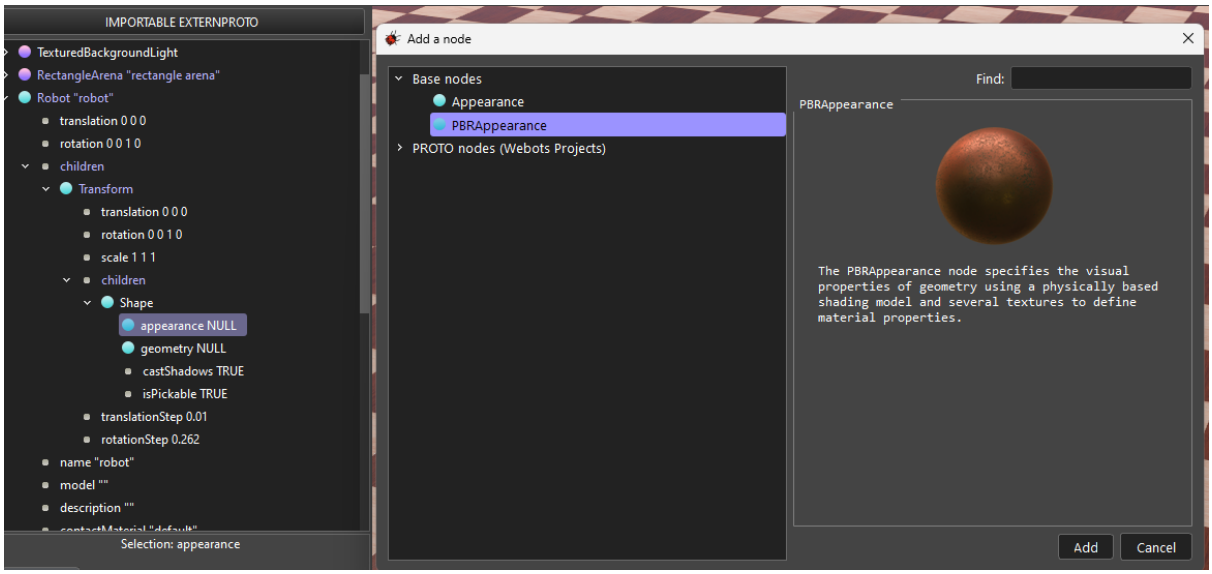

Рисунок 3.8 – Визначення візуальних властивостей геометрії

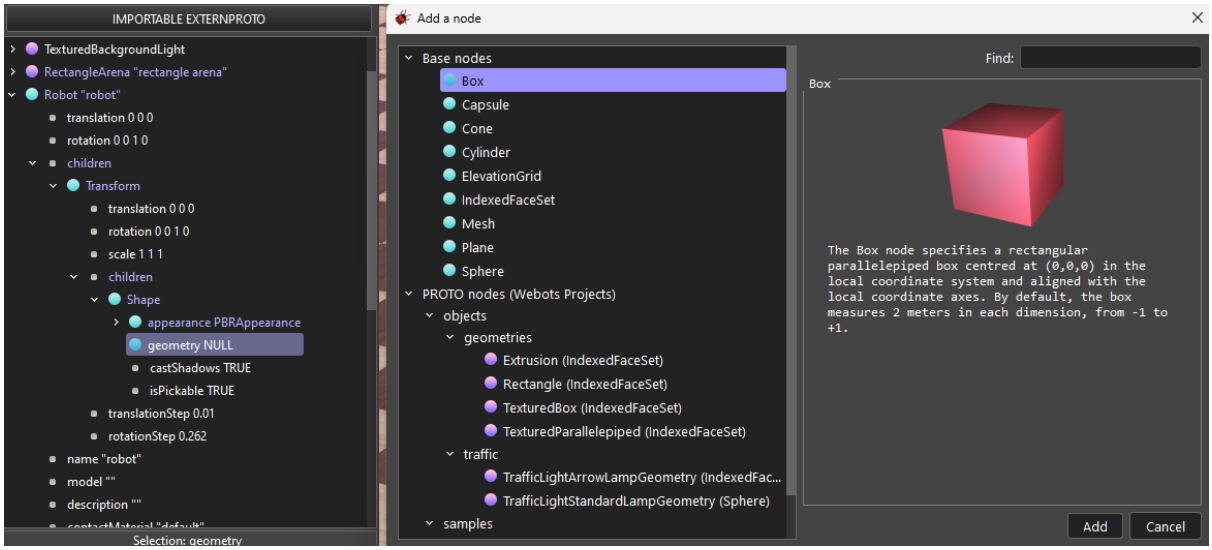

Рисунок 3.9 – Додавання геометрії тіла робота

Змінимо розміри корпусу робота (рис. 3.10), а саме за осями x, y та z на параметри (0,1; 0,16; 0,04) м.

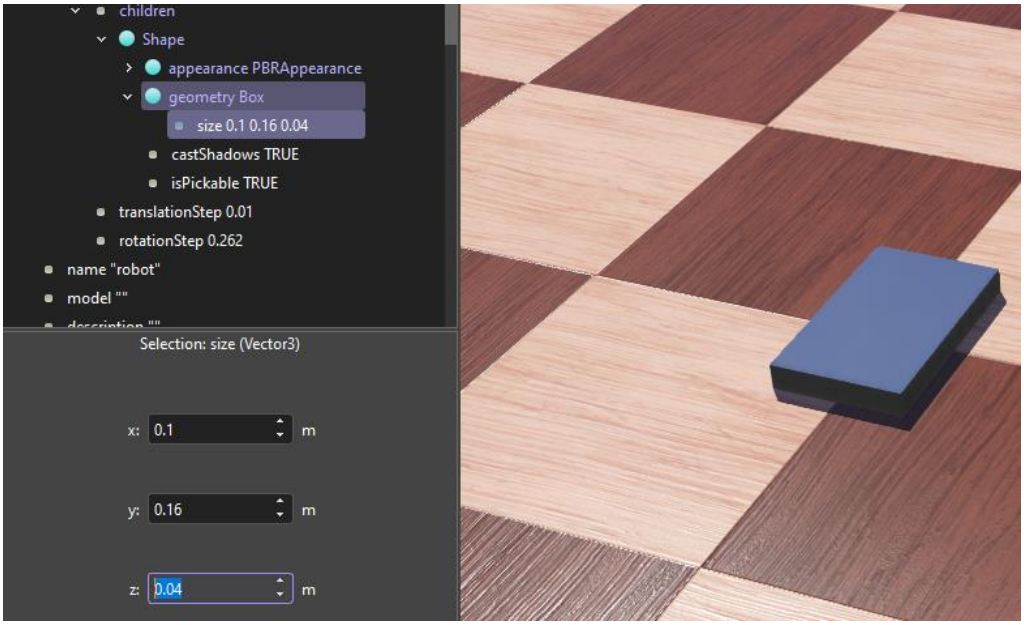

Рисунок 3.10 – Зміна геометричних параметрів корпусу робота

Далі потрібно змінити кольори робота на (0.6, 0.9, 0.5), його шорсткість (roughness) на 0 та металевість (metalness) теж на 0 (рис. 3.11), а також додати назву новому вузлу (наприклад, Body).

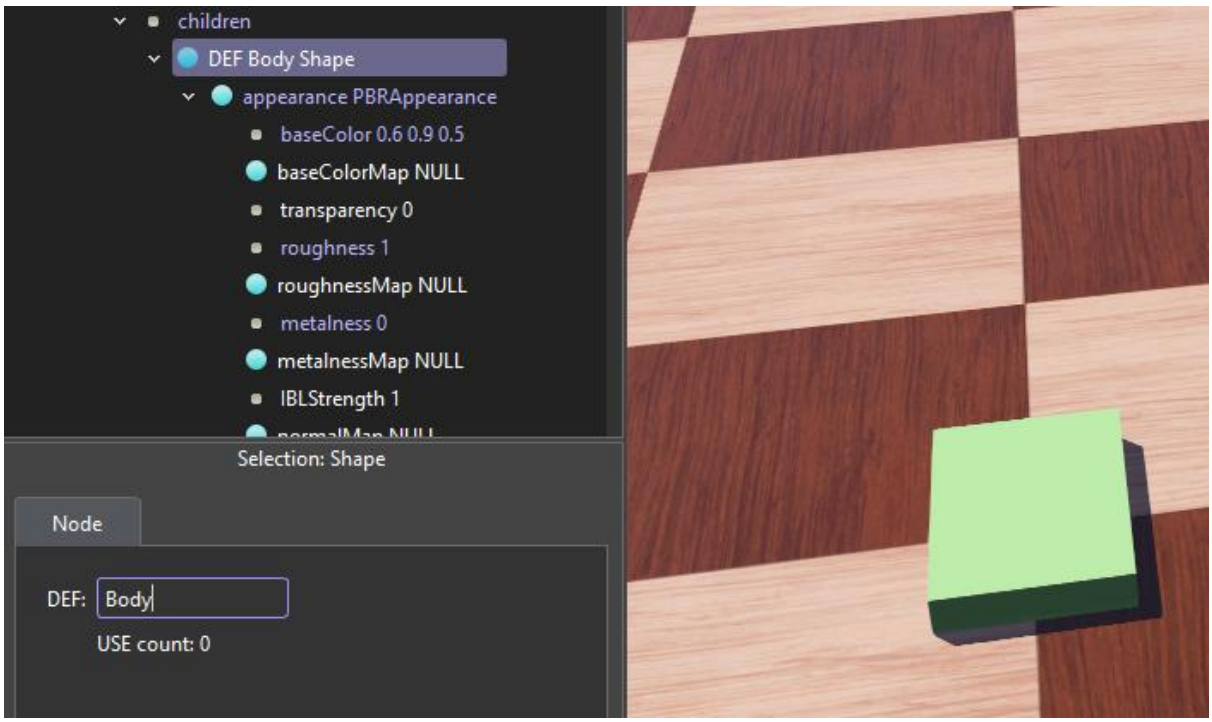

Рисунок 3.11 – Візуальні параметри тіла робота

Далі потрібно додати тілу колізію для можливості праці з нею (рис. 3.12) та обрати вузол, створений на попередньому етапі, а також додати фізику тілу (рис. 3.13).

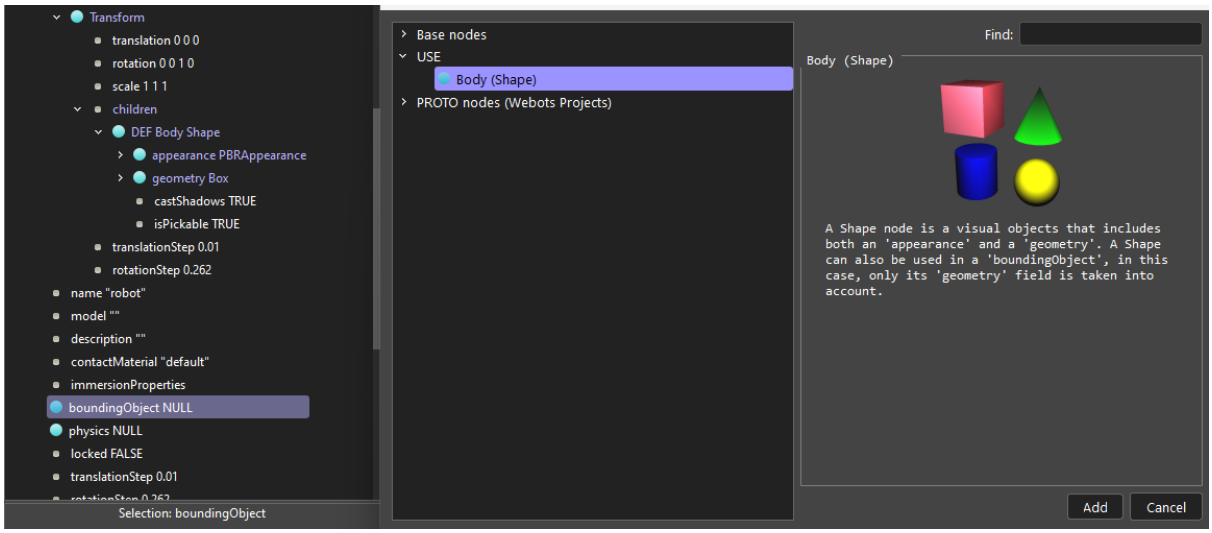

Рисунок 3.12 – Додавання колізії

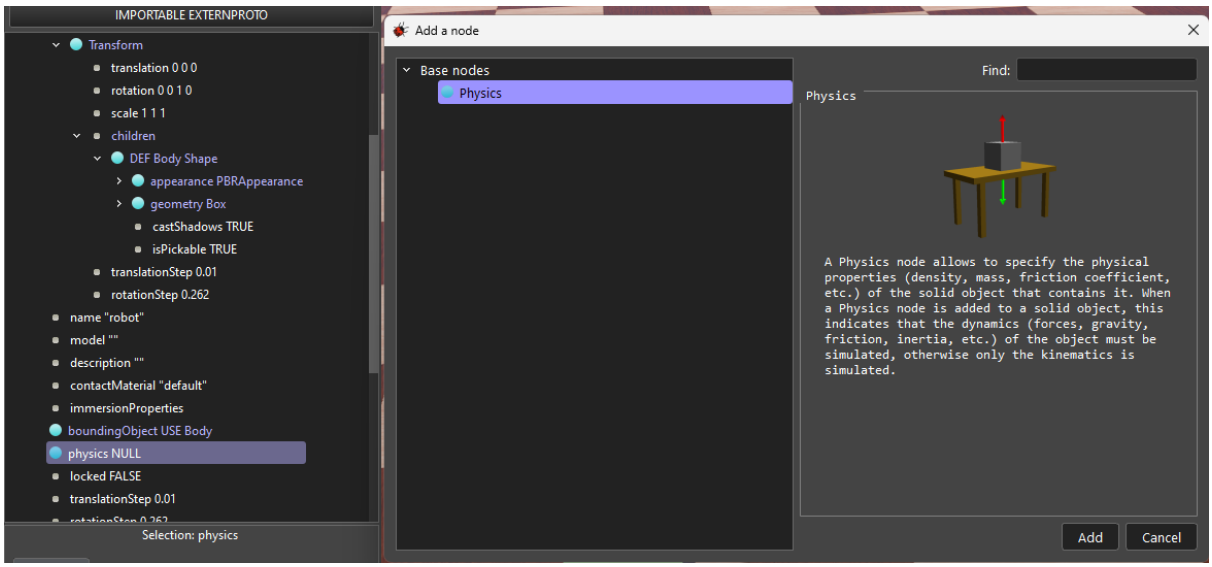

Рисунок 3.13 – Додавання фізики тілу

Після цього потрібно підняти корпус робота вище за віссю z на 0,04 м, щоб додати колеса (рис. 3.14).

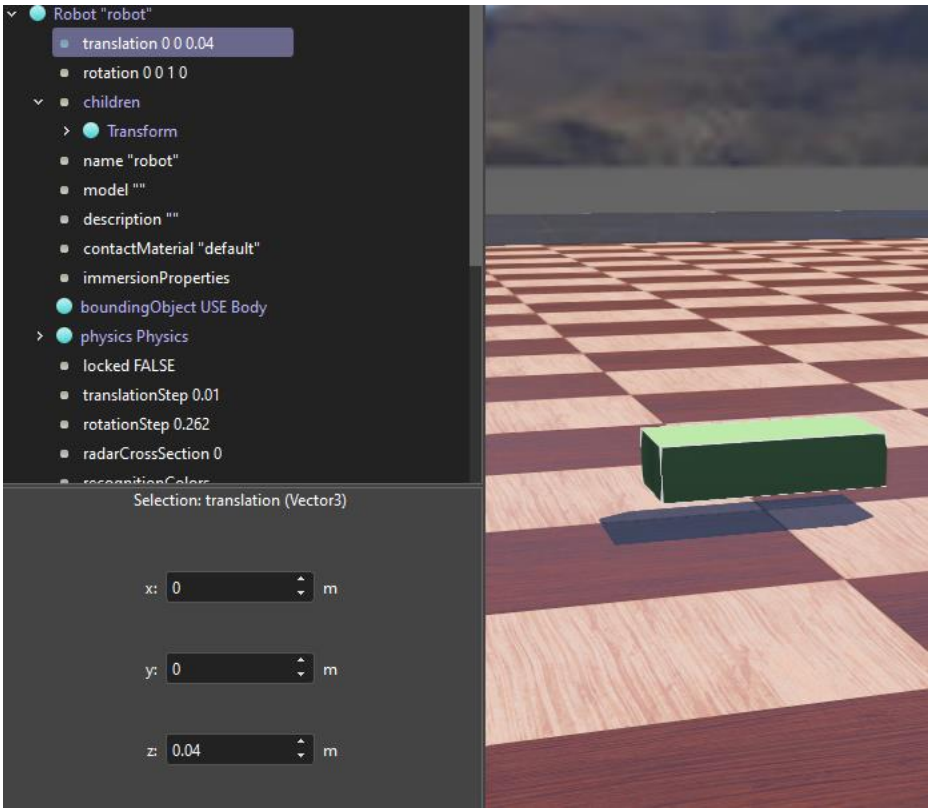

Рисунок 3.14 – Піднімання тіла робота вище

Створимо колесо. Для цього додамо новий дочірній елемент HingeJoint – для симуляції обертового руху (рис. 3.15).

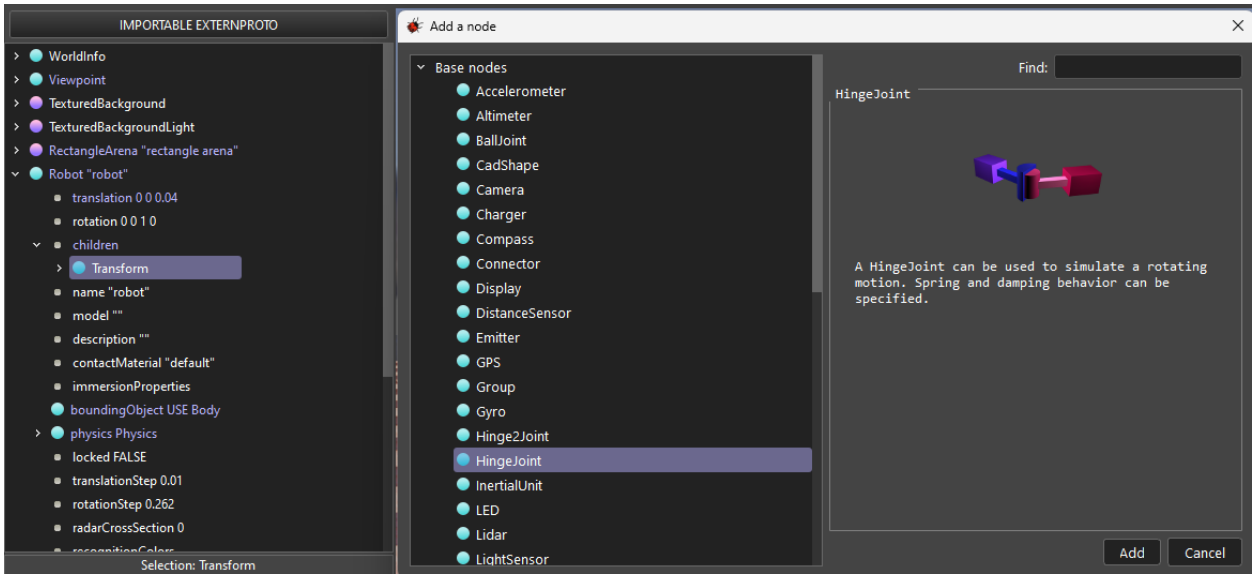

Рисунок 3.15 – Створення елементу обертового руху

Оберемо наступні параметри (рис. 3.16):

– jointParameters – HingeJointParameters (параметри вузла);

– device – RotationalMotor (мотор для обертового руху) з ім’ям left motor;

– endPoint – Solid (для представлення об'єктів у змодельованому середовищі).

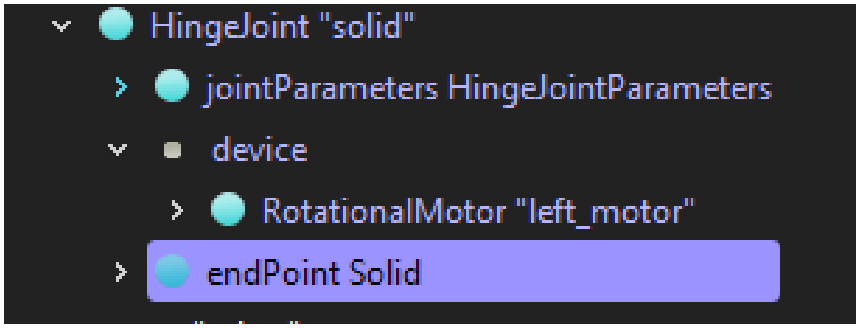

Рисунок 3.16 – Параметри для створення колеса

Для endPoint дочірнім елементом оберемо Shape (рис. 3.17).

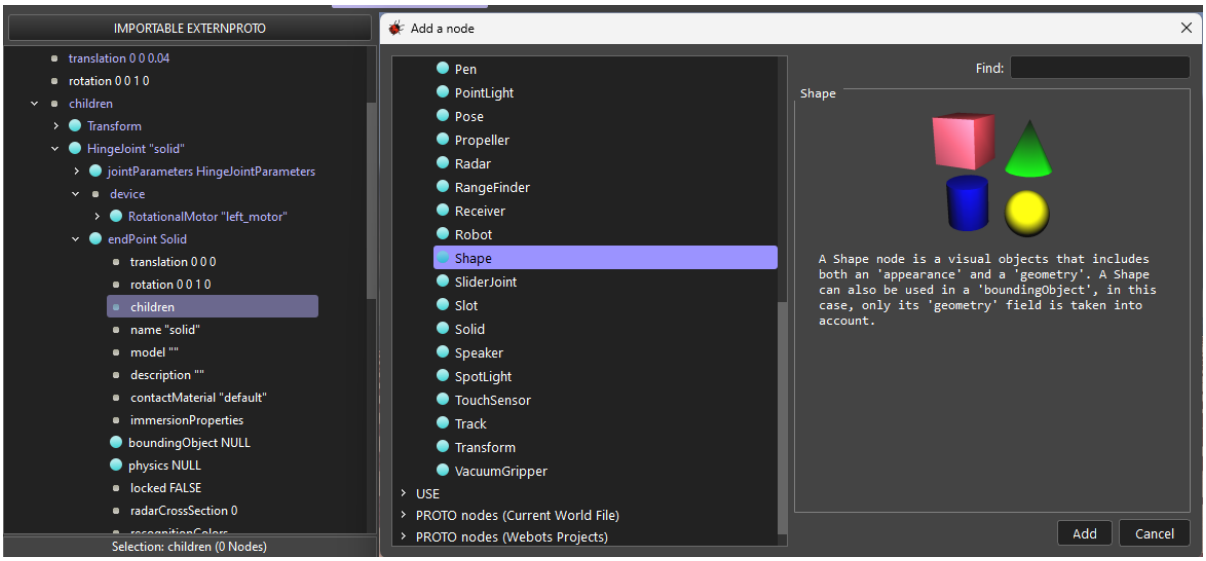

Рисунок 3.17 – Створення дочірнього елементу форми для колеса

Для форми Shape встановимо параметри appearance – PBRAppearance geometry – Cylinder (рис. 3.18).

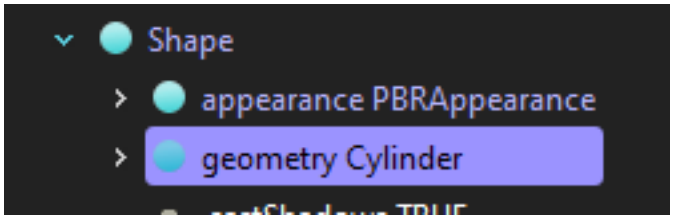

Рисунок 3.18 – Налаштування Shape

В геометрії колеса змінити параметри висоти та радіуса на 0,02 м.

Далі потрібно перемістити колесо в координати (0,05 м; -0,05 м; -0,02 м), а також повернути на 1,57 рад (90 градусів) уздовж осі y, а також назвати вузол як wheel (рис. 3.19).

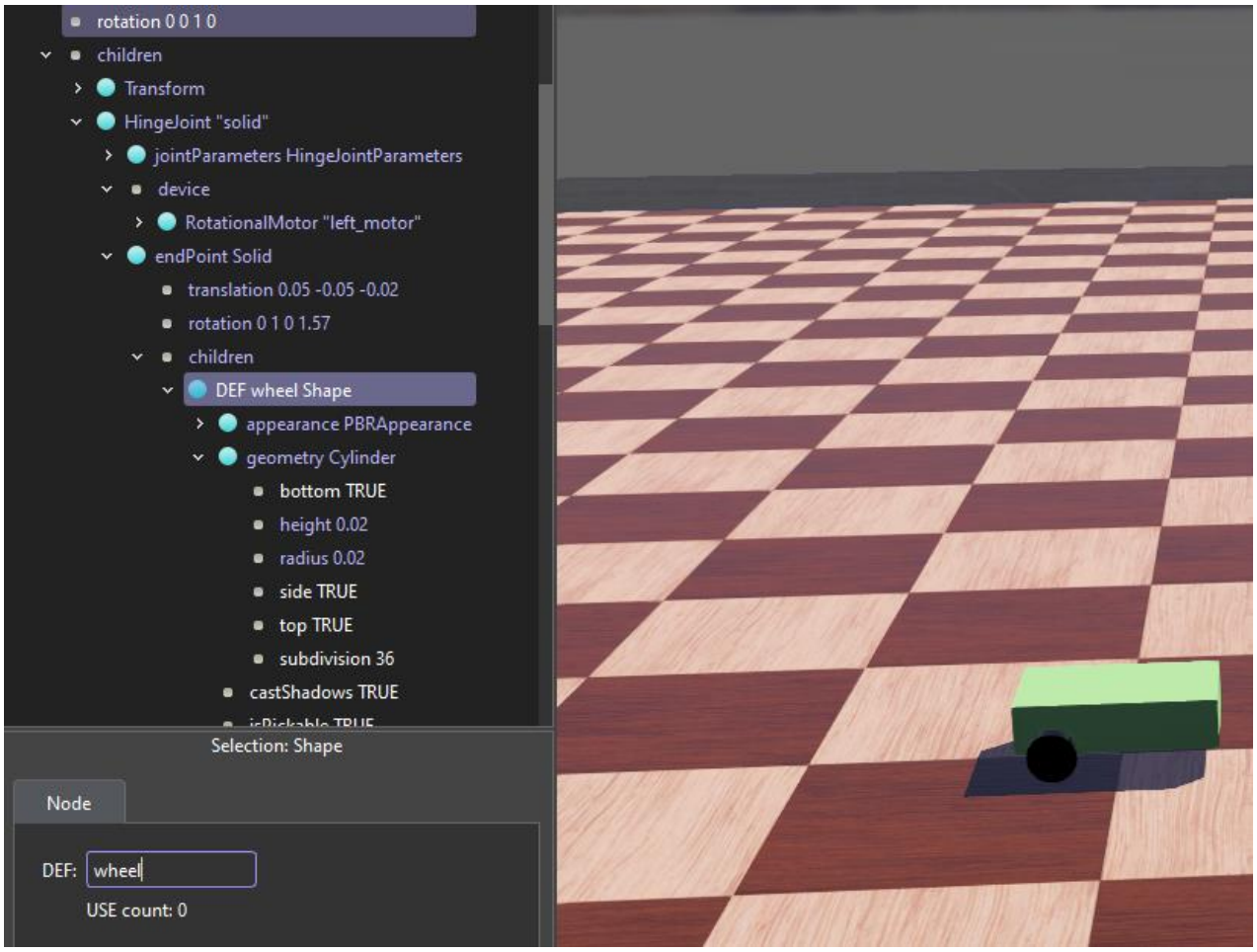

Рисунок 3.19 – Створення першого колеса

Далі додати колізію за допомогою створеної форми колеса та фізику (рис. 3.20).

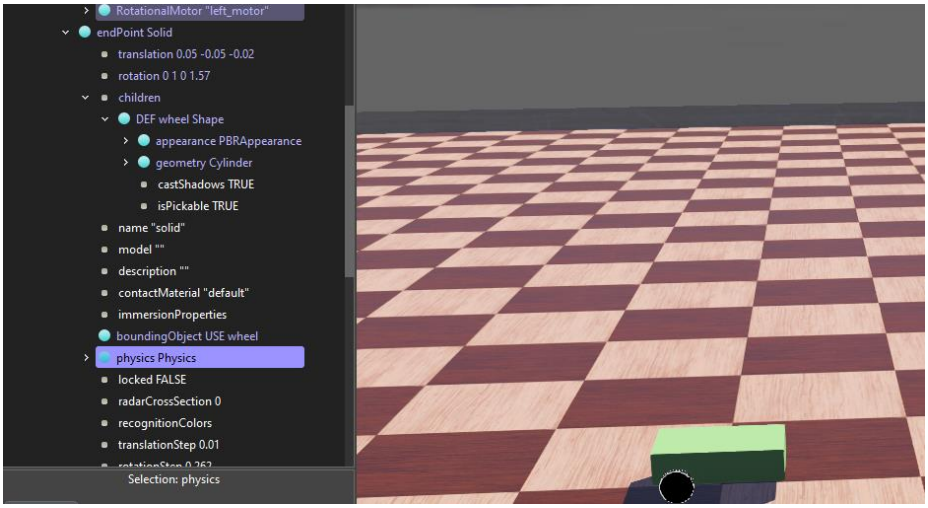

Рисунок 3.20 – Додавання колізії

Також потрібно додати anchor – прив’язку: це поле вказує позицію прив’язки, тобто точку, через яку проходить вісь шарніра. Разом із полем осі, успадкованим від вузла JointParameters, поле прив’язки визначає вісь повороту петлі унікальним чином – такими самими як і параметри переміщення колеса (рис. 3.21).

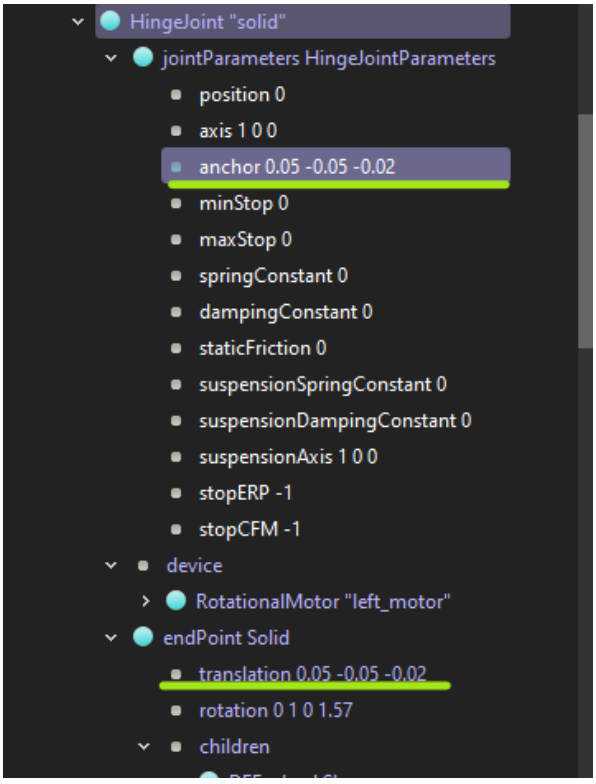

Рисунок 3.21 – Додавання прив’язки елементу

Аналогічним чином створюється і друге колесо, лише будуть змінені деякі параметри (рис. 3.22), а саме координати за віссю x, а також прив’язка.

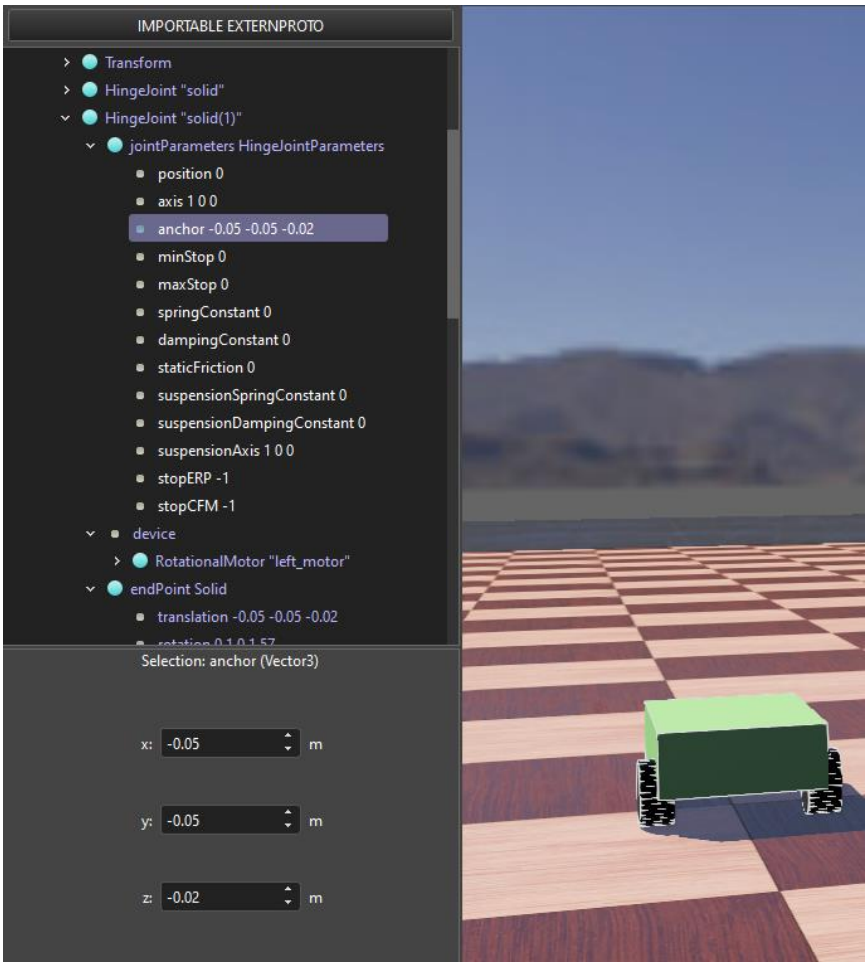

Рисунок 3.22 – Параметри другого колеса

Додамо третє підтримуюче колесо з іншої сторони робота. Для цього створимо вузол BallJoint для симуляції обертового руху за трьома осями координат (рис. 3.23).

Далі додамо властивості для цього колеса, а саме BallJointParameters, JointParameters та JointParameters для кожного з налаштувань (рис. 3.24) та встановимо позиціонування за осями x та y (рис. 3.25).

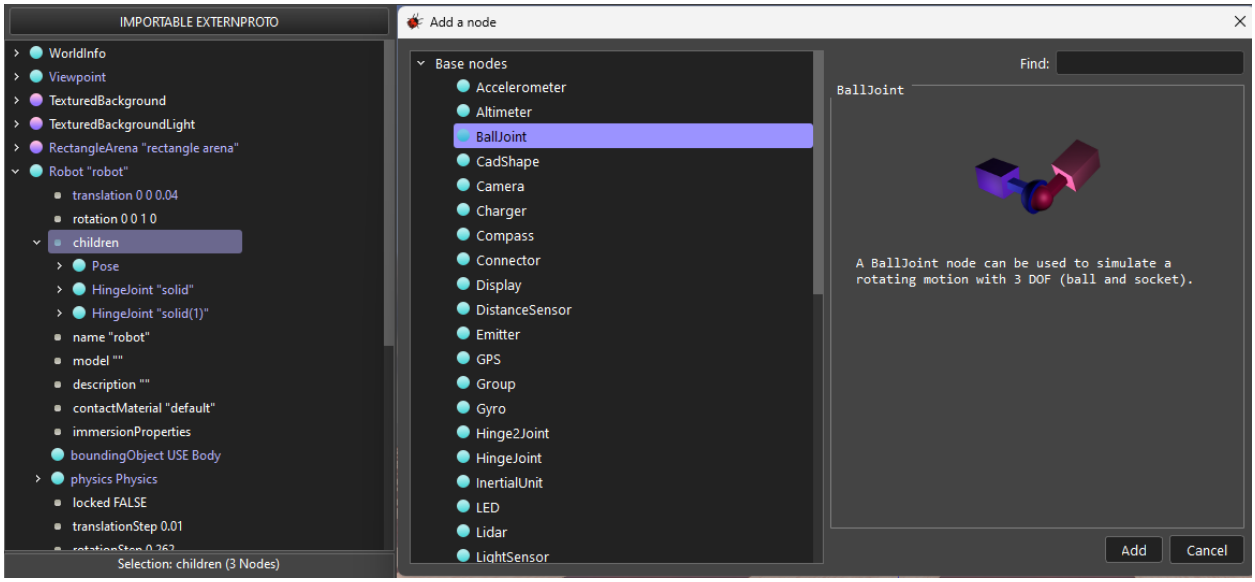

Рисунок 3.23 – Створення вузлу для третього колеса

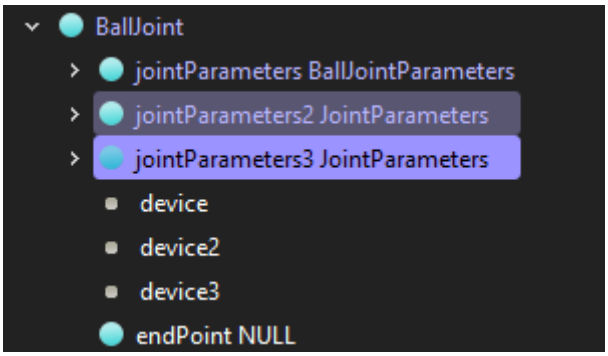

Рисунок 3.24 – Додавання властивостей колесу

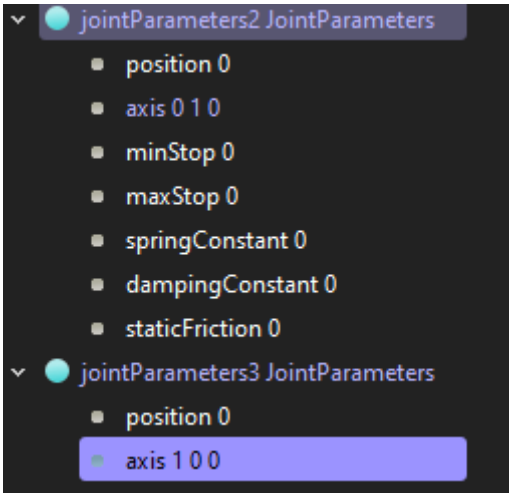

Рисунок 3.25 – Налаштування роботи за осями x та y

У якості параметра endpoint додамо вузол Solid, у якості його дочірнього елементу оберемо Shape, георметрія – сфера (Sphere), що й буде третім підтримуючим колесом (рис. 3.26).

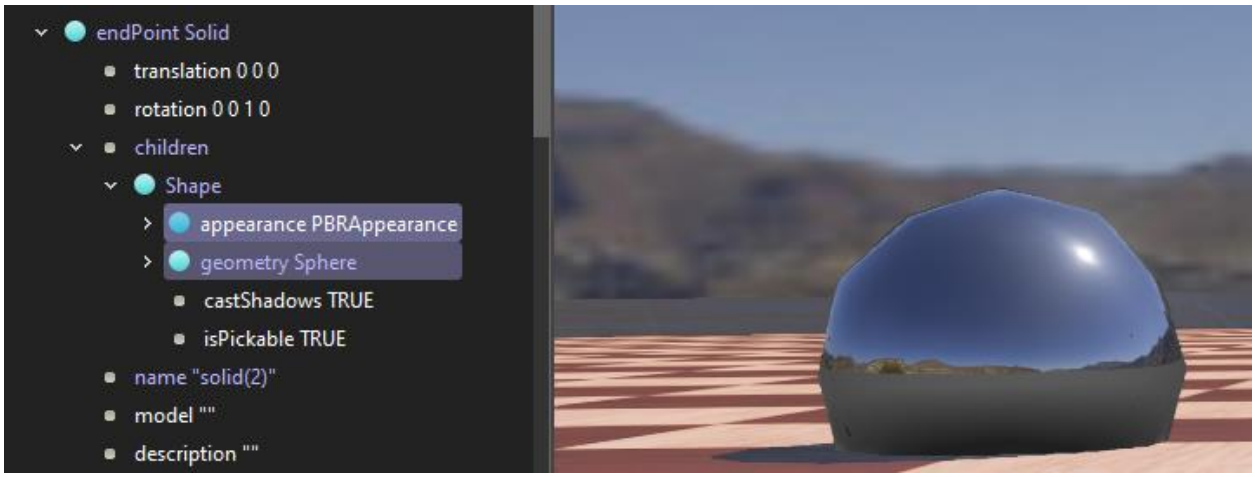

Рисунок 3.26 – Додавання вузлу для третього колеса

Змінемо відповідні параметри сфери для колеса (рис. 3.27).

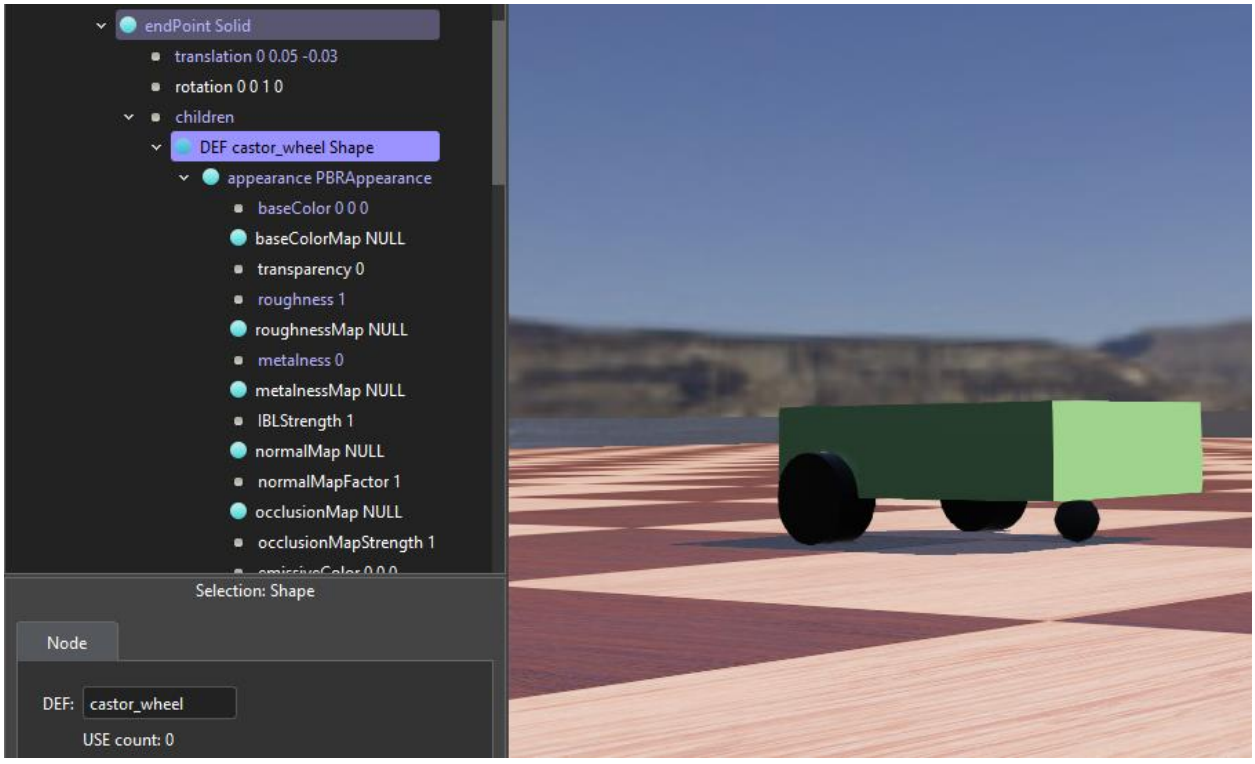

Рисунок 3.27 – Зміна параметрів колеса

Цими параметрами є:

– DEF castor_wheel – назву елементу колеса;

– geometry – radius = 0,01 м;

– appearance – baseColor (0; 0; 0) – чорний колір колеса;

– roughness та metallnes (згладжування та металевість)– 1 та 0 відповідно;

– translation – (0 м; 0,05 м; -0,03 м) – переміщення за осями.

Для всього колеса додамо відчутність (використовуємо той же вузол з назвою castor_wheel) та фізику (рис. 3.28).

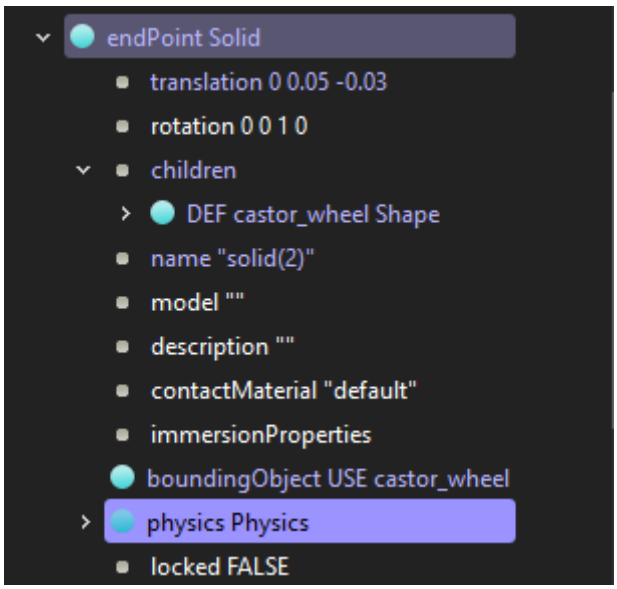

Рисунок 3.28 – Додавання відчутності та фізики

Також потрібно додати точку обертання (anchor) – таку саму як і параметри translation (рис. 3.29).

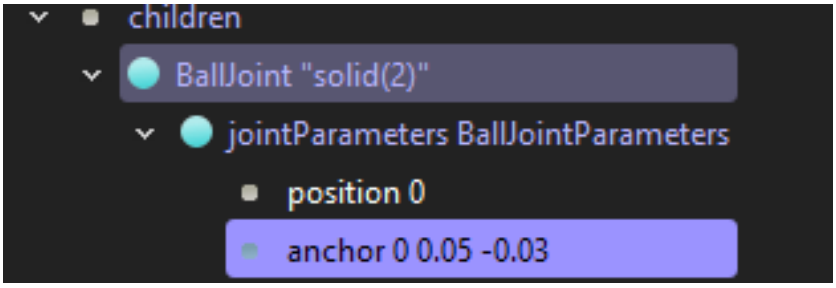

Рисунок 3.29 –Додавання точки обертання

Додамо кожух навколо колеса. Для цього створимо ще один вузол типу Solid, в якості дочірнього елементу до нього оберемо Shape. Для вузлу Shape оберемо в якості параметрів геометрії циліндр з параметрами висоти та радіусу 0,01 м та 0,01 м відповідно, а також у якості appearance – PBRAppearance параметри згладжування та металевості 1 та 0 відповідно, параметри кольору (0,5; 0,5; 0,5), що відповідає сірому кольору. Також для вузлу Solid змінити параметри переміщення на (0 м; 0,05 м; -0,025 м) (рис. 3.30).

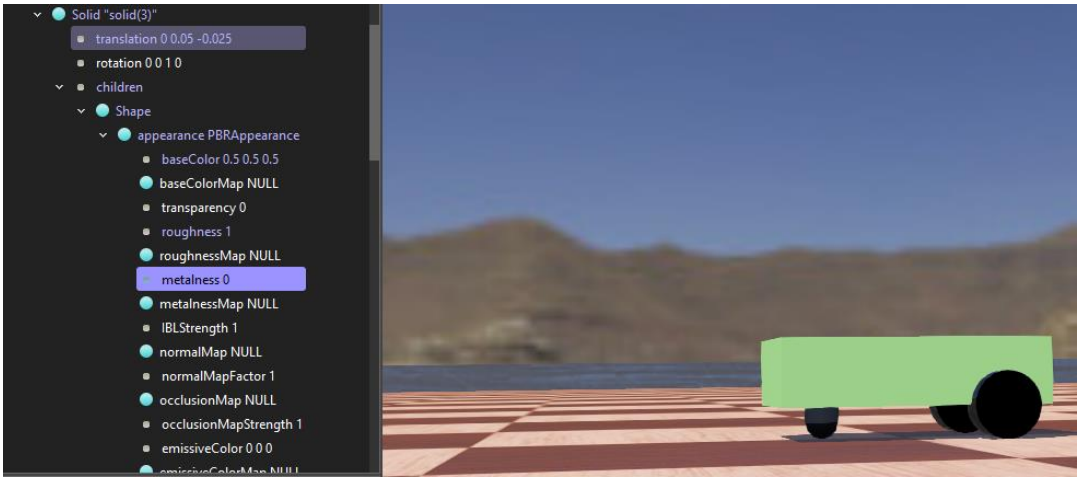

Рисунок 3.30 – Створення кожуху для підтримуючого колеса

3.2.2 Створення простого контролеру для перевірки роботи моделі

Для керування роботом потрібно створити новий контролер мовою Python (рис. 3.31-3.33).

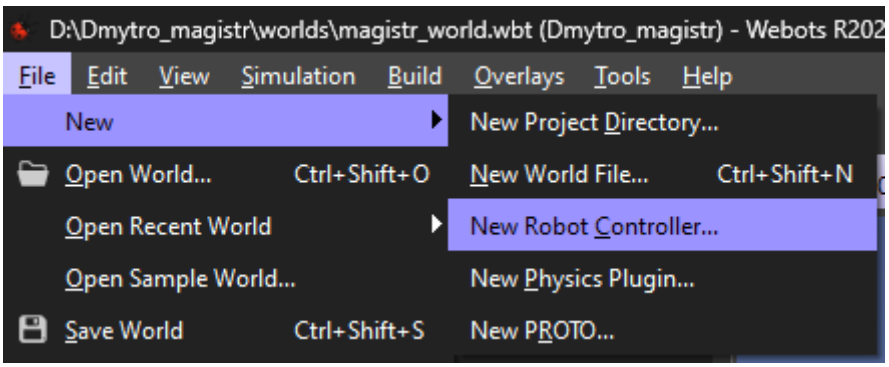

Рисунок 3.31 – Створення контролеру через меню

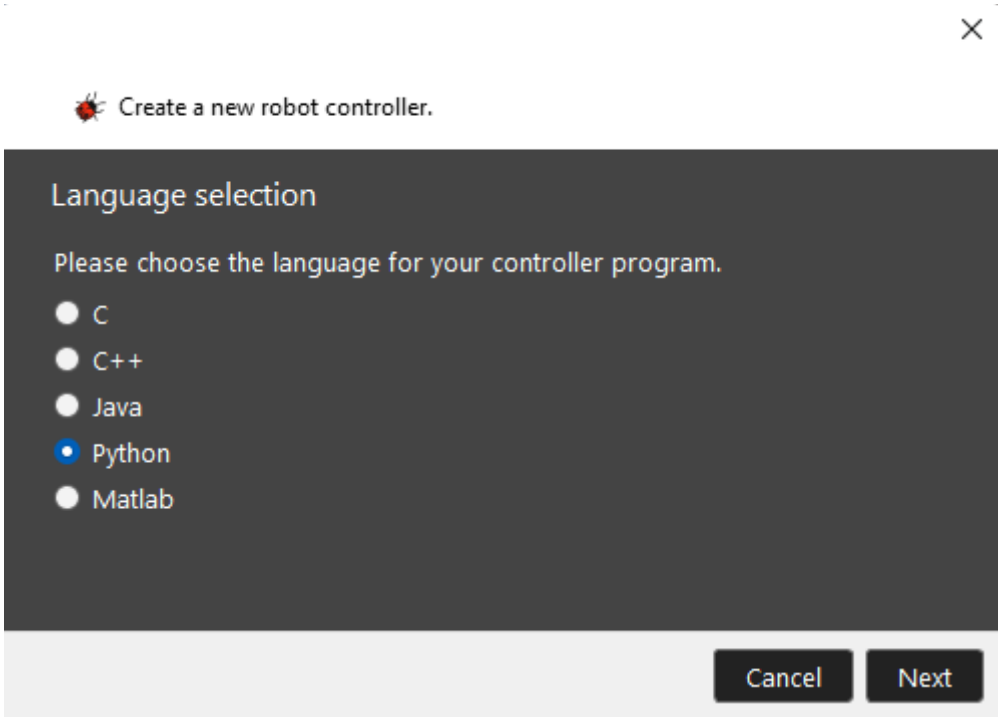

Рисунок 3.32 – Вибір мови програмування

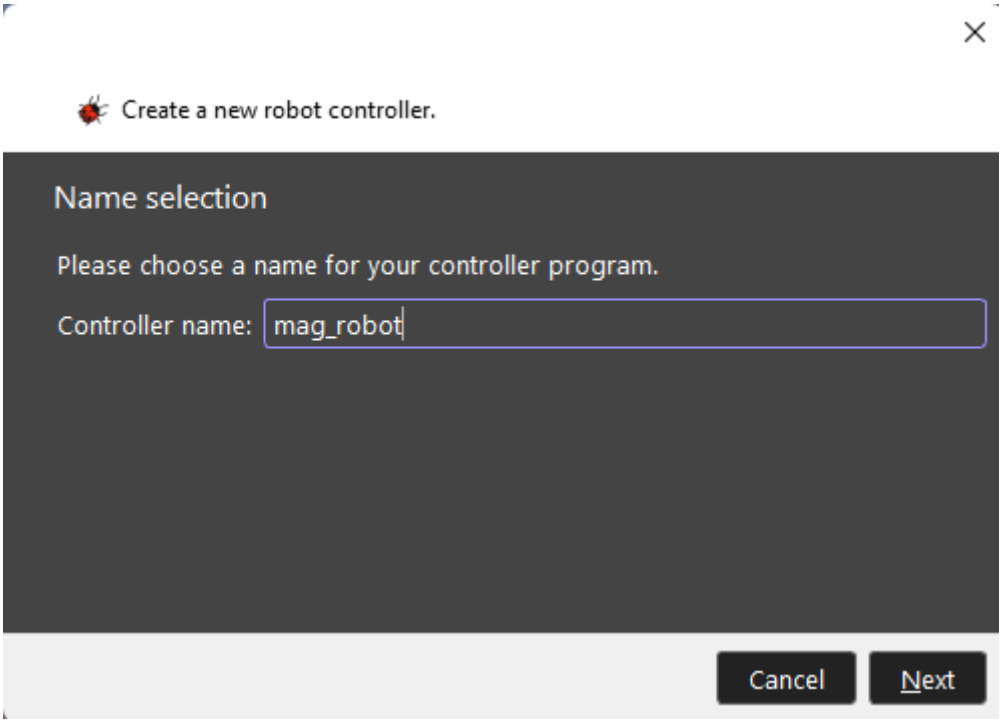

Рисунок 3.33 – Введення імені контролеру

Далі потрібно підключити бібліотеку для керування роботами:

```
from controller import Robot
```

Після цього створити об’єкт робота та задати основні змінні (часовий крок у програмі та максимальну швидкість робота):

```
robot = Robot()
timestep = 64
max_speed = 6.28
```

Далі треба отримати доступ до створених двигунів:

```
lm = robot.getDevice("left_motor")
rm = robot.getDevice("right_motor")
```

Встановити початкове значення двигунів (в даному випадку воно не є важливим, тому значення нескінченності):

```
lm = robot.getDevice("left_motor")
rm = robot.getDevice("right_motor")
```

Встановити початкові швидкості для кожного з моторів:

```
lm.setVelocity(0.0)
rm.setVelocity(0.0)
```

А далі у нескінченому циклі задати швидкості переміщення робота:

```
while robot.step(timestep) != -1:
    lm.setVelocity(6.28*0.5)
    rm.setVelocity(6.28*0.3)
```

Далі потрібно цей контролер підключити до робота. Для цього потрібно його обрати для вузлу робота, натиснувши кнопку select та знайшовши відповідний збережений контролер (рис. 3.34).

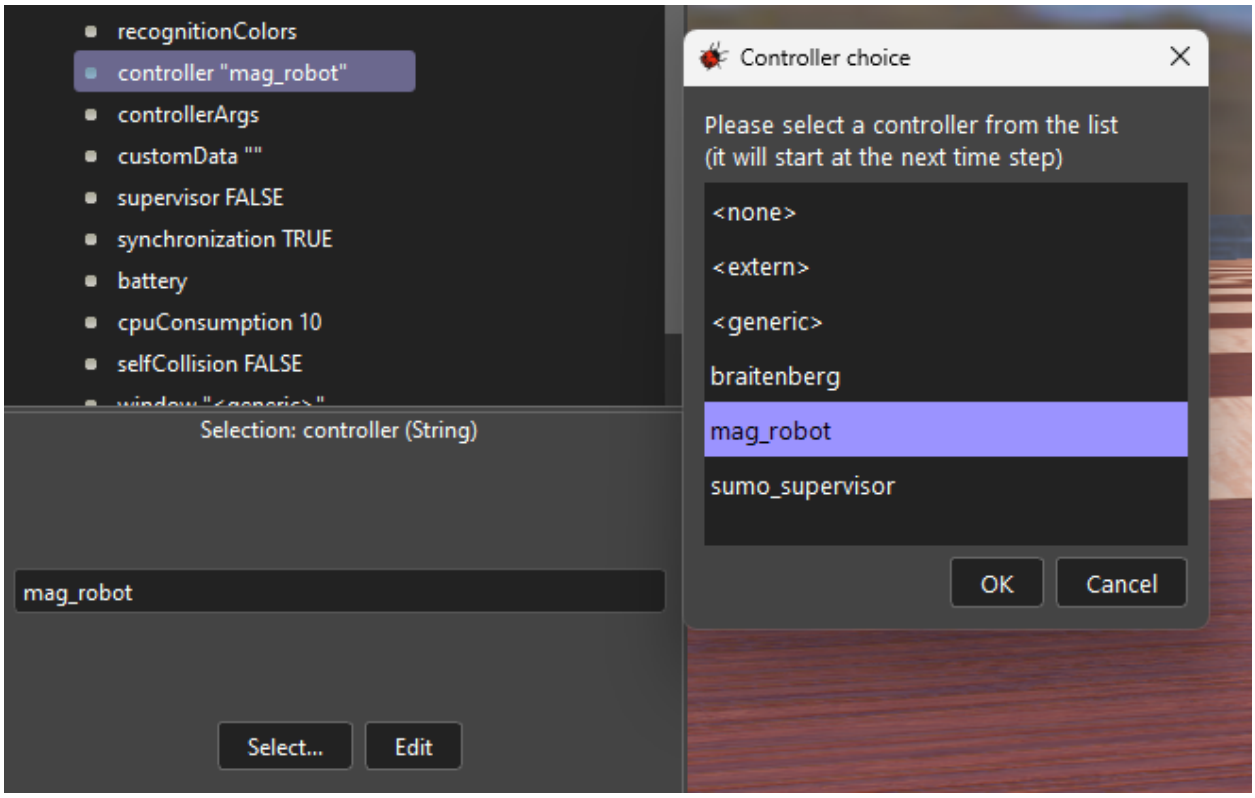

Рисунок 3.34 – Підключення контролеру

3.2.3 Додавання датчиків лінії до моделі робота

Створимо новий вузол для датчика відстані DistanceSensor (рис. 3.35), для нього дочірнім елементом оберемо Solid, для якого, в свою чергу, Shape. Для Shape оберемо appearance – PBRAppearance параметри згладжування та металевості 1 та 0 відповідно, параметри кольору (0,0; 0,0; 0,7), що відповідає синьому кольору. У якості геометрії оберемо капсулу (Capsule) висотою 0,01 м радіусом 0,002 м (рис. 3.36).

Для того щоб побачити напрямок дії променів сенсора, оберемо відповідний параметр з меню View – Optional Rendering – Distance Sensor Rays (рис. 3.37), він буде показаний сірим кольором.

Потім потрібно встановити правильний напрямок променів (рис. 3.38), де спочатку налаштовано напрямок променів, а потім вже й напрямок капули.

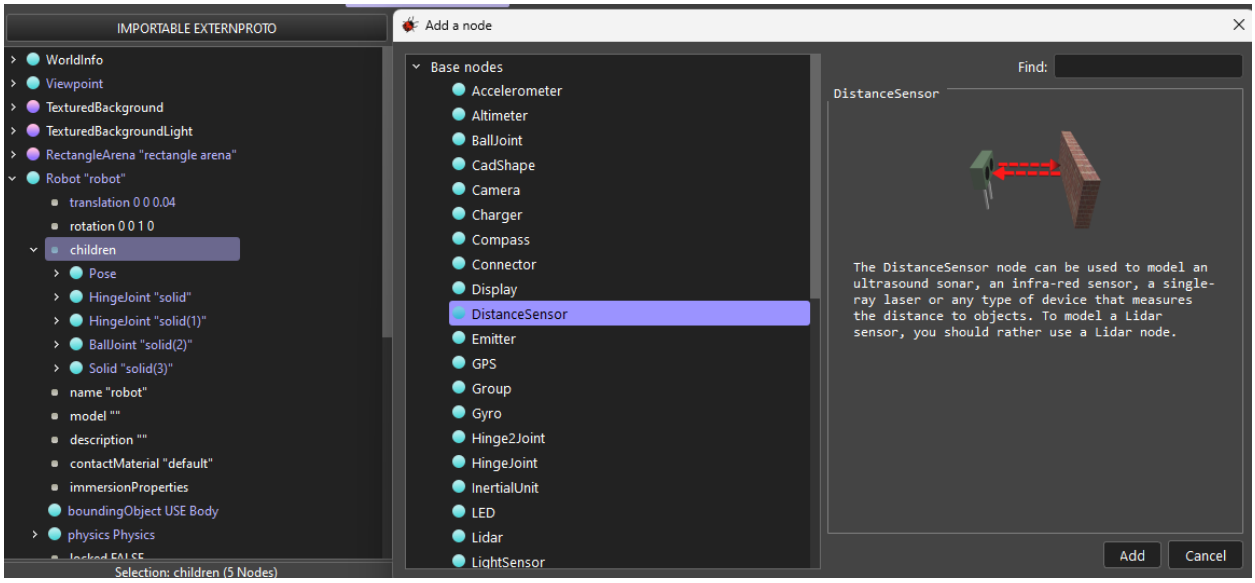

Рисунок 3.35 – Створення сенсору

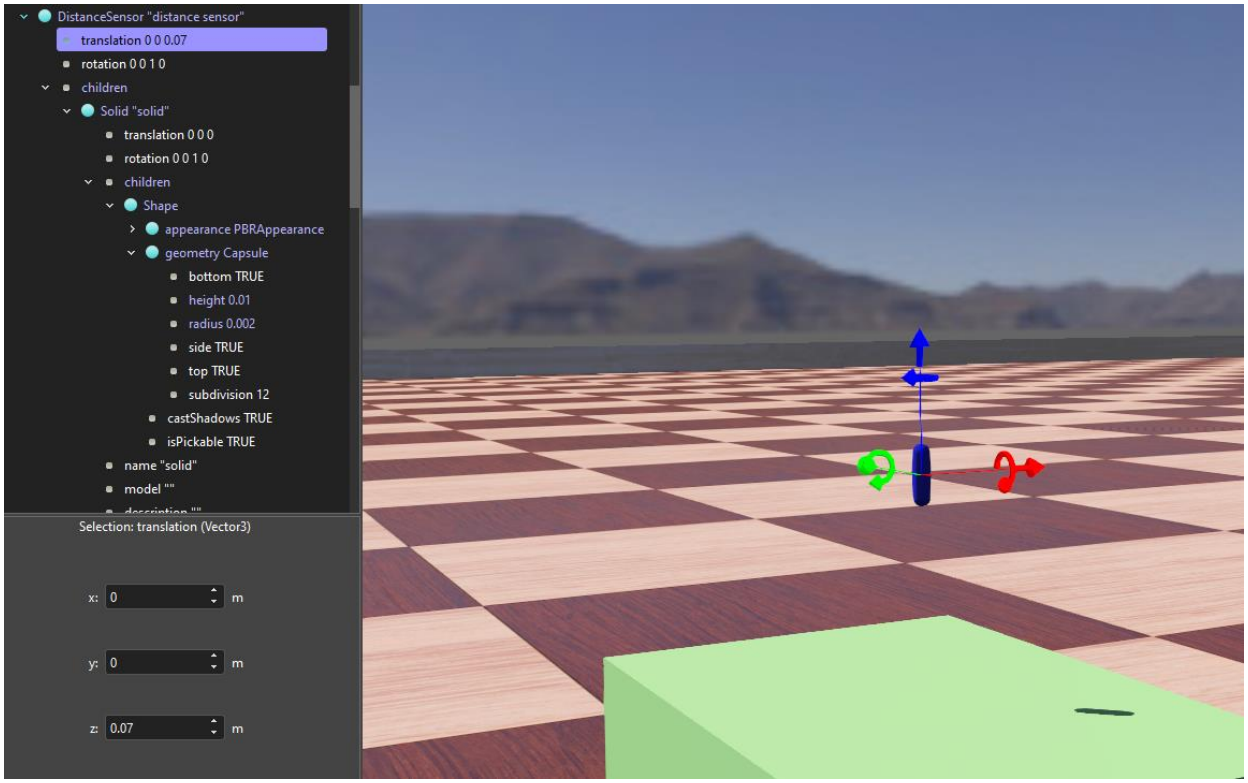

Рисунок 3.36 – Зовнішній вигляд сенсора

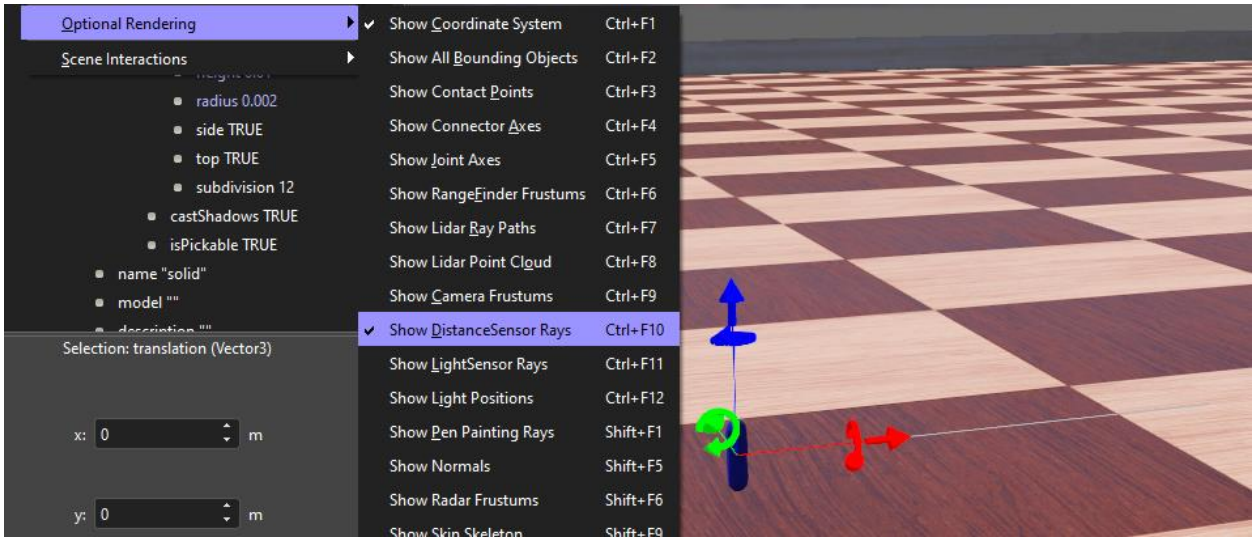

Рисунок 3.37 – Промені сенсора

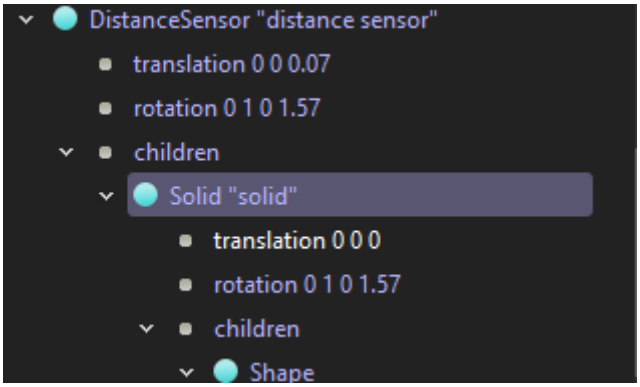

Рисунок 3.38 – Параметри повороту для датчика

Далі змінимо положення датчика у роботі на його передню частину (рис. 3.39), а саме змінити координати датчика на (0 м; -0,075 м; -0,02 м).

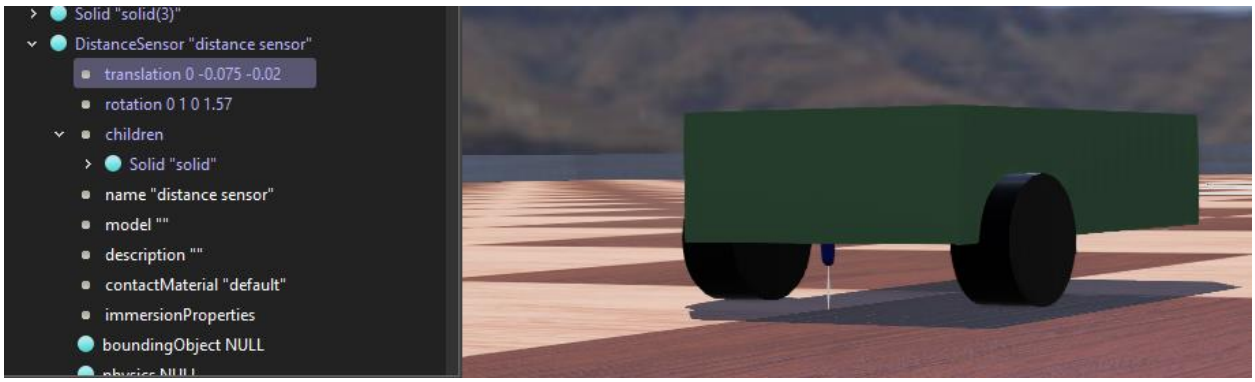

Рисунок 3.39 – Зміна положення датчика

Встановимо налаштування датчика як інфра-червоного (рис. 3.40), а також назву датчика для програмування (рис. 3.41).

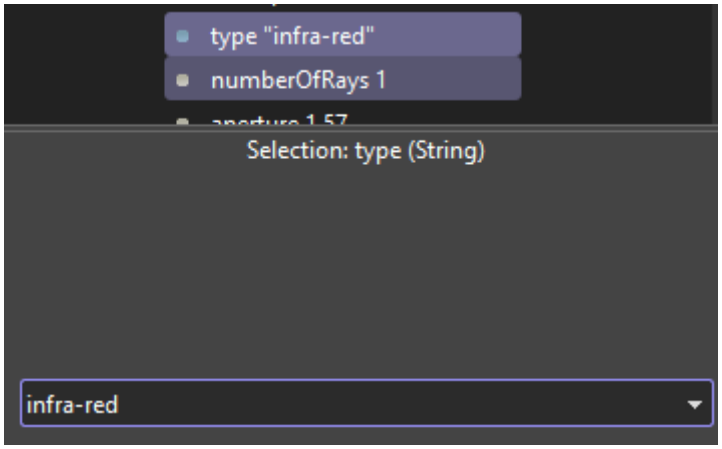

Рисунок 3.40 – Зміна типу датчика

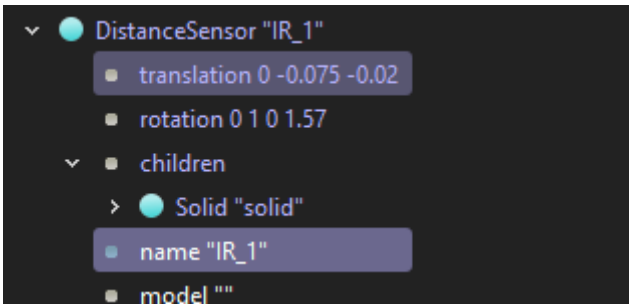

Рисунок 3.41 – Зміна назви датчика

Для перевірки роботи цього сенсору, змінемо код контролеру, а саме створимо об’єкт датчика, а також додамо крок часу у нашому віртуальному просторі:

```
IR = robot.getDistanceSensor("IR_1")
IR.enable(timestep)
```

Далі створимо змінну, яка буде зберігати показання датчиків і виводити їх у консоль:

```
k= IR.getValue()
```

```
print (k)
```

Результат отримання показань з датчиків представлено на рис. 3.42.

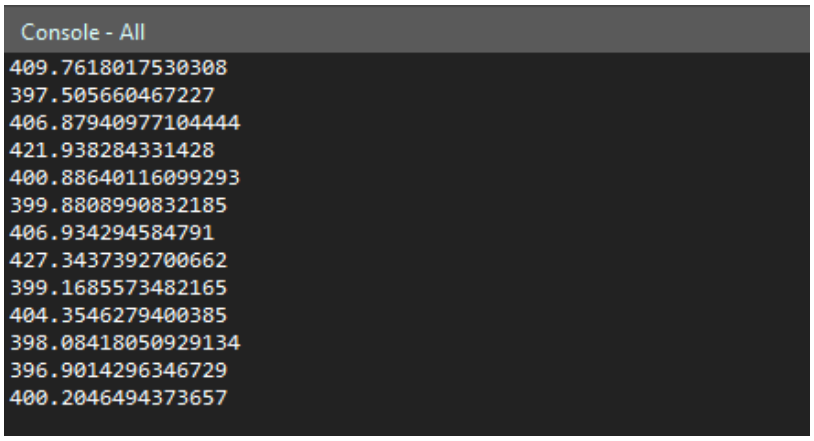

Рисунок 3.42 – Отримання даних з датчику

Для робота було вирішено використати 8 датчиків. Для цього можна скопіювати вже готовий, змінивши його координати та назви датчиків на (IR_1; IR_2; IR_3; IR_4; IR_5; IR_6; IR_7; IR_8), а також координати за x у проміжку від 0,035 м до -0,035 м з кроком у 0,01 м.

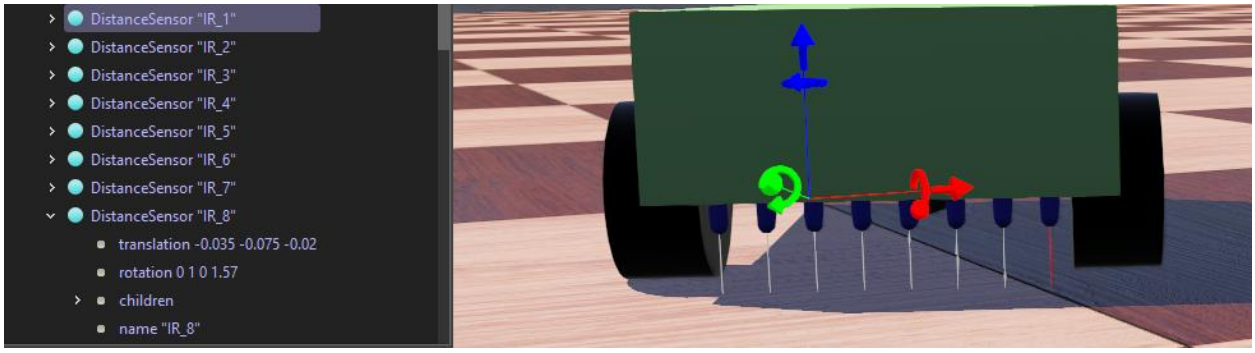

Рисунок 3.43 – Розташування датчиків робота

Створимо пустий масив датчиків:

```
sensors = []
```

Далі створимо масив з іменами датчиків, з яких будемо отримувати інформацію:

```
names = ["IR_1","IR_2","IR_3","IR_4","IR_5","IR_6","IR_7","IR_8"]
```

А також пустий масив для отримання чисельної інформації датчиків:

```
panel = [0,0,0,0,0,0,0,0]
```

Для програмної ініціалізації датчиків використаємо цикл:

```
for i in range (0,8):
    sensors.append(robot.getDistanceSensor(names[i]))
    sensors[i].enable(timestep)
```

Також використаємо цикл для отримання інформації з датчиків:

```
for i in range (0,8):
        panel[i]=sensors[i].getValue()
        print(panel)
```

Результат отримання інформації з датчиків показано на рис. 3.44.

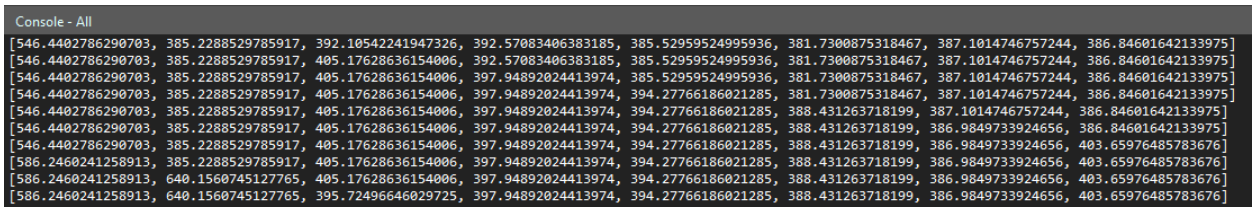

Рисунок 3.44 – Отримання інформації з датчиків

3.3 Створення програмного забезпечення для руху лінією за допомогою ПІД-регулятора

Змінимо програму керування, що була представлена у попередньому підрозділі.

Створимо п’ять змінних:

```
previous_error = 0.0
kp = 0
ki = 0
kd = 0
Integral = 0.0
```

Кожна з цих змінних відповідає за окремий параметр:

– previous_error – змінна що використовується для збереження помилки, яка сталася під час останнього виконання (під час обчислення диференційної складової);

– kp – пропорційна складова ПІД-регулятора;

– ki – інтегральна складова ПІД-регулятора;

– kd – диференційна складова ПІД-регулятора;

– Integral – змінна для збереження суми помилок переміщення. На кожному кроці обчислення поточна помилка додається до цієї змінної (Integral= Integral+ error).

Створимо функцію для отримання даних з ІЧ-сенсорів та запису їх у масив reading:

```
def getReading():
    for i in range (0,8):
        if int(sensor[i].getValue())>512:
```

```
            reading[i]=0
        else:
            reading[i]=1
```

У симуляторі будь-яке значення, отримане з ІЧ-сенсора є аналоговим, тому потрібно перетворити його у цифрове, а масив reading зберігає значення лише 0 та 1, а не float.

Так як використовується 8 сенсорів, тому було додано цикл для читання з кожного з 8 сенсорів, що потрібні на кожному етапі виконання алгоритму ПІД-регулятора.

Коли функція getReading викликається один раз, отримуються дані з усіх 8 сенсорів і записані у масив reading у цифровому форматі. Порогова функція для перетворення у цифровий формат є 512, так як дані, зчитані з ІЧ-сенсора завжди у проміжку від 0 до 1023.

Якщо дані, зчитані з сенсора більше 512, тоді у масив reading записується 0 у відповідну комірку масиву. Це порогове значення має бути вибрано таким чином, щоб можна було чітко ідентифікувати чорне та біле.

У поточних умовах освітлення, що використовуються в цьому проекті, це значення приблизно дорівнює 512.

Якщо необхідно зайти чорну лінію, то потрібно змінити відповідну логіку в функції getReading.

Завжди датчики над лінією повинні видавати одиницю, а інші нуль. На практиці порогові значення слід вибирати відповідно до умов освітлення, оскільки джерела світла та інші джерела тепла теж випромінюють ІЧ-промені.

Наступною є функція, яка реалізує ПІД-регулятор та його алгоритм:

```
def PID():
    error = 0
    coefficient=[-4000,-3000,-2000,-1000,1000,2000,3000,4000]
    #error=coefficient[0]*reading[0]+coefficient[1]*reading[1]....
```

```
for i in range(0,8):
    error+=coefficient[i]*reading[i]
P=kp*error
I=Integral+(ki*error)
D=kd*(error-previous_error)
correction=(P+I+D)/1000
l_speed=5+correction
r_speed=5-correction
if l_speed<0.0 : l_speed=0
if l_speed>10.0 : l_speed=10.0
if r_speed<0.0 : r_speed=0
if r_speed>10.0 : r_speed=10.0
lm.setVelocity(l_speed)
rm.setVelocity(r_speed)
print(l_speed, r_speed, reading)
return I, error
```

У будь-якому виді ПІД-алгоритм базується на миттєвій помилці. Тому нам потрібно створити змінну (error = 0) для збереження цієї помилки.

Оскільки значення помилки завжди може змінюватися, і воно використовується лише один раз, нам не потрібно виділяти глобальну змінну.

Далі створимо масив значень (coefficient=[-4000,-3000,-2000,-1000,1000,2000,3000,4000]) – це вагові коефіцієнти кожного з датчиків. Кожна значення з сенсора перемножується з відповідним значенням коефіцієнта.

Якщо, наприклад, значення з сенсорів [0,0,1,1,1,0,0], то помилка буде обрахована як:

$$error = 0 \cdot (-4000) + 0 \cdot (-3000) + 1 \cdot (-2000) + 1 \cdot (-1000) + 1 \times$$
$$\times (1000) + 0 \cdot (2000) + 0 \cdot (3000) + 0 \cdot (4000) = -2000.$$

Для розрахунку помилки з усіх 8 ІЧ-датчиків було використано цикл:

```
for i in range(0,8):
    error+=coefficient[i]*reading[i]
```

Далі потрібно знайти пропорційну, інтегральну та диференційні частини:

```
P=kp*error
I=Integral+(ki*error)
D=kd*(error-previous_error)
```

Якщо ми побудуємо графік цієї величини помилки, ми отримаємо дискретний графік, а амплітуда цього графіка є аналогом пропорційного коефіцієнта. Величина повороту сильно залежить від пропорційної складової.

Інтегральна та диференційна складові мають незначний вплив на налаштування регулятора.

Інтегральна складова відповідає за повторювані помилки.

Диференціальний коефіцієнт схожий на конденсатор у ланцюзі випрямляча – згладжує коливальний рух робота.

Фактично вихідний сигнал ПІД-регулятора сильно залежить від пропорційного коефіцієнта, незначно залежить від диференціального коефіцієнта та дуже слабо залежить від інтегрального коефіцієнта.

Пропорційний фактор вносить коригування курсу, диференціальний коефіцієнт виконує точні налаштування, а інтегральний коефіцієнт робить уточнюючі налаштування.

У таких програмах, таких як слідування лінією, зазвичай не використовують інтегральний коефіцієнт, і це означає, що він дорівнює нулю, або можна вказати дуже мале значення. Тому що під час повороту двигуна робота, наприклад на 90 градусів, він повинен зупиняти мотор на 85-88

градусах і повільно за рахунок інерції допрацьовувати, а у випадку використання великого значення інтегральної складової, мотор буде зупинятися на 90 градусах і продовжувати рух на 2-3 градуси, що може викликати додаткові помилки. Але у випадку руху лінією ця складова не буде значно впливати на рух робота.

Сума корекції, або коєфіцієнт визначається наступним чином:

```
correction=(P+I+D)/1000
```

Шляхом ділення від 1000, зміна помилки зменшена до діапазону швидкості двигуна.

Максимальна швидкість моторів дорівнює 10 рад/с, тому середня швидкість тут 5 рад/с. Якщо ми не зменшуємо швидкість двигуна, вона завжди збільшується. Середня швидкість двигуна становить 5 рад/с. Коли кількість корекції додається, швидкість змінюється поперемінно навколо 5 рад/с, а критичні значення тут 0 рад/с та 10 рад/с.

Набір коефіцієнта, який використано, є лінійним. Це означає, що різниця між двома послідовними способами є однаковою або рівномірною і дорівнює 1000.

Після обчислення корекції ми додаємо та видаляємо її від значень швидкостей моторів:

```
l_speed=5+correction
r_speed=5-correction
```

Наступні рядки призначені для контролю переповнення. Якщо швидкість менше за 0, то вона є 0, якщо більше максимального 10, то 10:

```
if l_speed<0.0 : l_speed=0
if l_speed>10.0 : l_speed=10.0
```

```
if r_speed<0.0 : r_speed=0
if r_speed>10.0 : r_speed=10.0
```

Далі записуємо швидкості в якості змінних до функцій керування моторами:

```
lm.setVelocity(l_speed)
rm.setVelocity(r_speed)
```

Також слід зауважити, що метод getReading повинен викликатися перед методом PID для коректної роботи програми.

Експериментальна частина з налаштуванням ПІД-регулятора буде наведена у розділі 4.

3.3 Моделювання роботи лідару у середовищі WeBots

Додамо до створеного робота датчик-лідар. Для цього потрібно створити новий вузол робота під назвою Lidar (рис. 3.45).

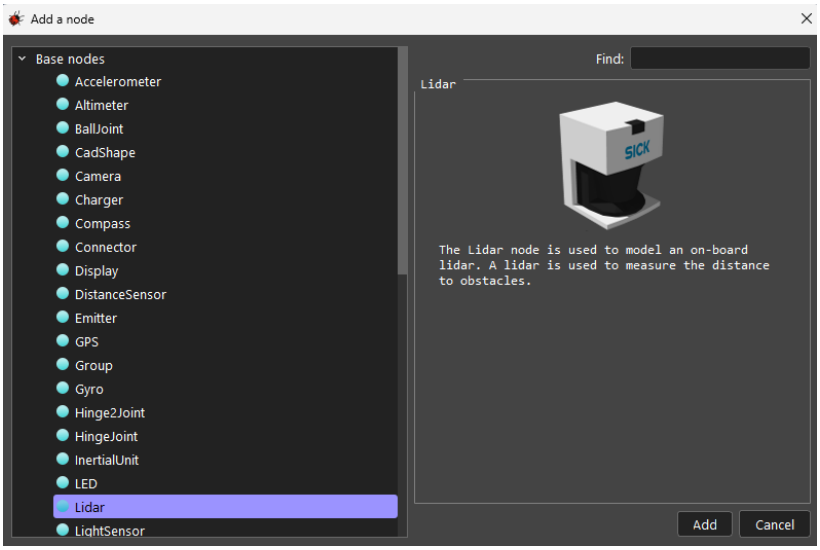

Рисунок 3.45 – Додавання лідару

Після цього додати дочірній до нього вузол Solid з дочірнім вузлом Shape. У якості геометрії обрати, наприклад, конус з параметрами нижнього радіусу і висоти (0,02 м; 0,02 м) відповідно. Параметр appearance – PBRAppearance має наступні налаштування (рис. 3.46):

– baseColor – (0; 0,4; 1);

– roughness – 1;

– metallness – 0.

Далі потрібно перенести сам датчик на передню сторону робота та відповідно його повернути.

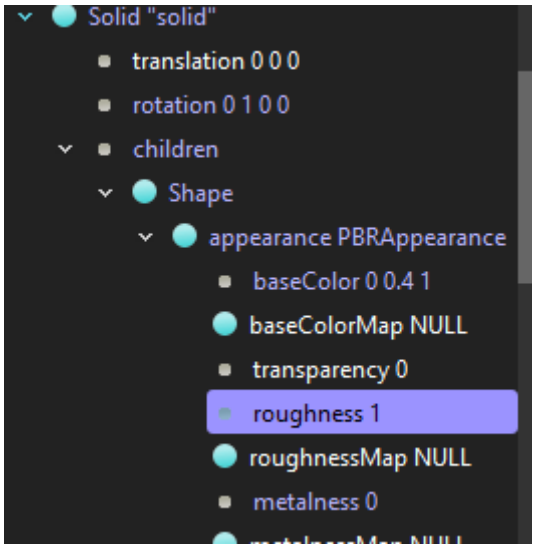

Рисунок 3.46 – Параметри зовнішнього вигляду лідару

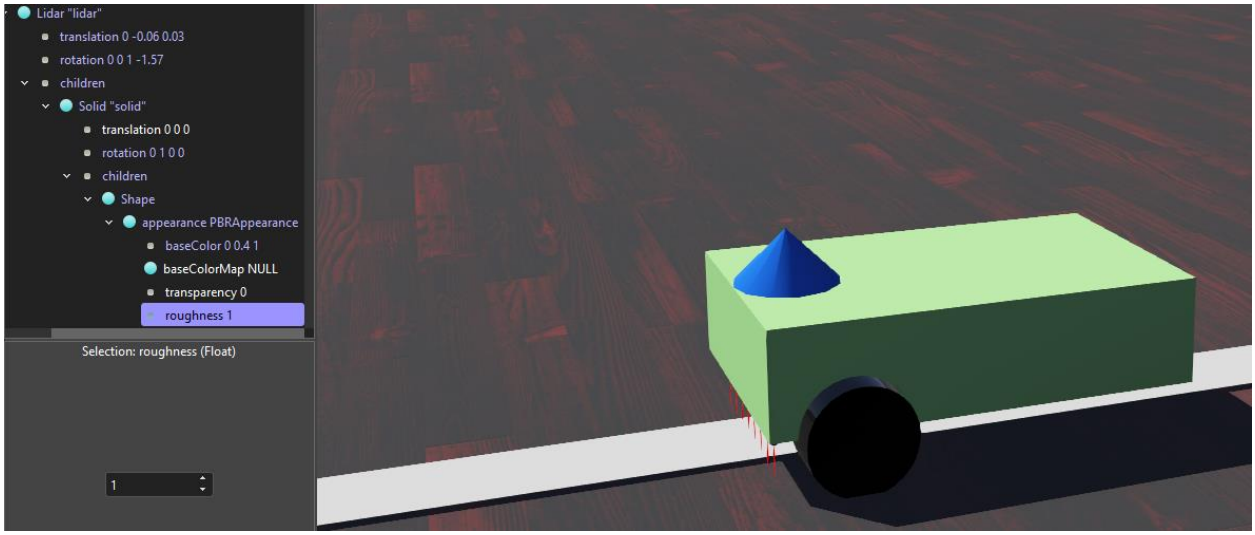

Рисунок 3.47 – Зовнішній вигляд доданого лідару

Параметрами які було обрано є:

– переміщення в координати (0 м; -0,06 м; 0,03 м);

– поворот за віссю z на –1,57 рад.

Для видимості променів лідара та його точок виставимо параметри із меню View – Optional Rendering (рис. 3.48).

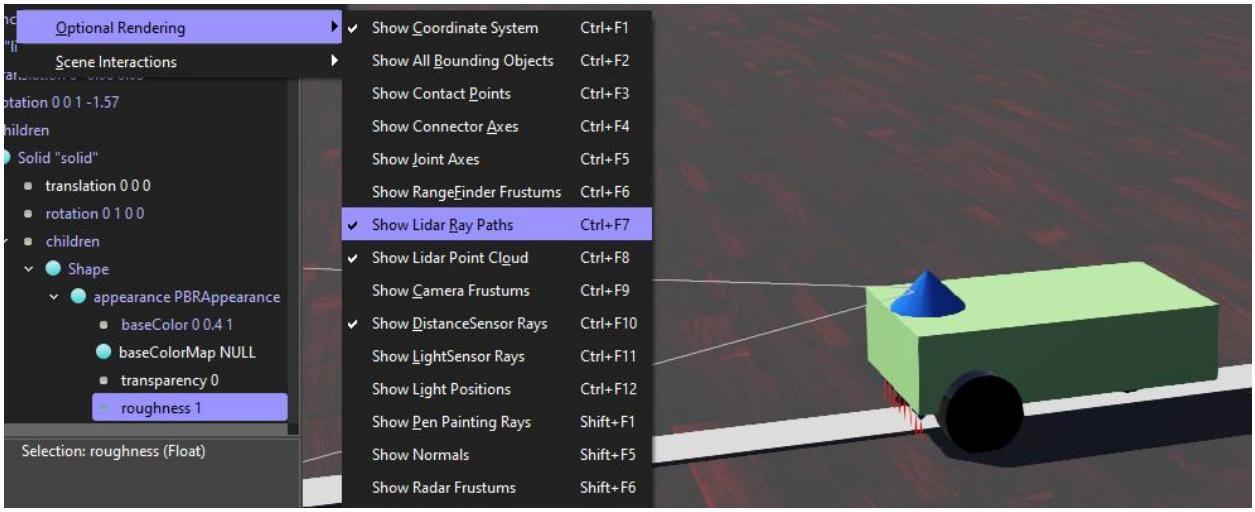

Рисунок 3.48 – Параметри показу променів лідару

Проведемо налаштування лідару (рис. 3.49), а саме:

горизонтальна роздільна здатність – 128;

поле зору – 1 рад.;

кількість шарів лідару – 1.

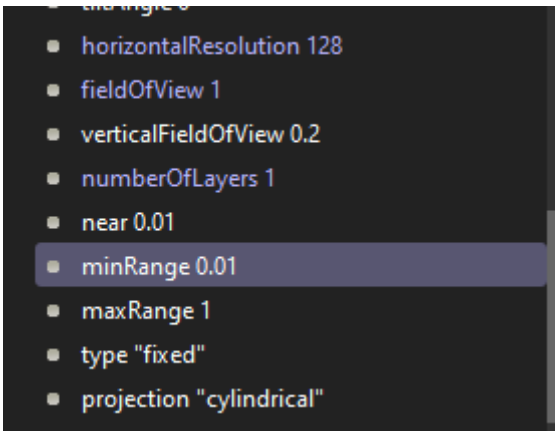

Рисунок 3.49 – Налаштування лідару

Після цього потрібно внести зміни до програми керування.

По-перше, потрібно створити об’єкт лідару, додати йому можливість праці з кроком часу симулятора, а також дозволити використовувати хмару точок лідару для візуального представлення:

```
lidar = robot.getDevice('lidar')
lidar.enable(timestep)
lidar.enablePointCloud()
```

В цилкі while robot.step(timestep) != -1 отримати дані з лідару, а саме 128 вимірювань, як було вказано у горизонтальній роздільній здатності (рис. 3.50):

```
range_image = lidar.getRangeImage()
print("{}".format(range_image))
```

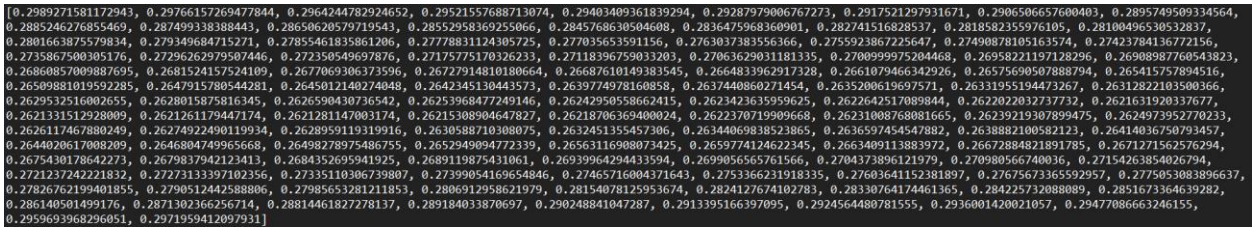

Рисунок 3.50 – Числові значення з лідару

Також далі додати умови для того, щоб робот зупинявся, якщо перешкода виявляється на відстані менше ніж 0,3 м, а іншому випадку робот рухається за лінією:

```
for ri in range_image:
        if ri < 0.3:
            lm.setVelocity(0.0)
            rm.setVelocity(0.0)
        else:
```

getReading()

Integral, previous_error=PID()

Результат виконання цієї частини програми представлено на рис. 3.51.

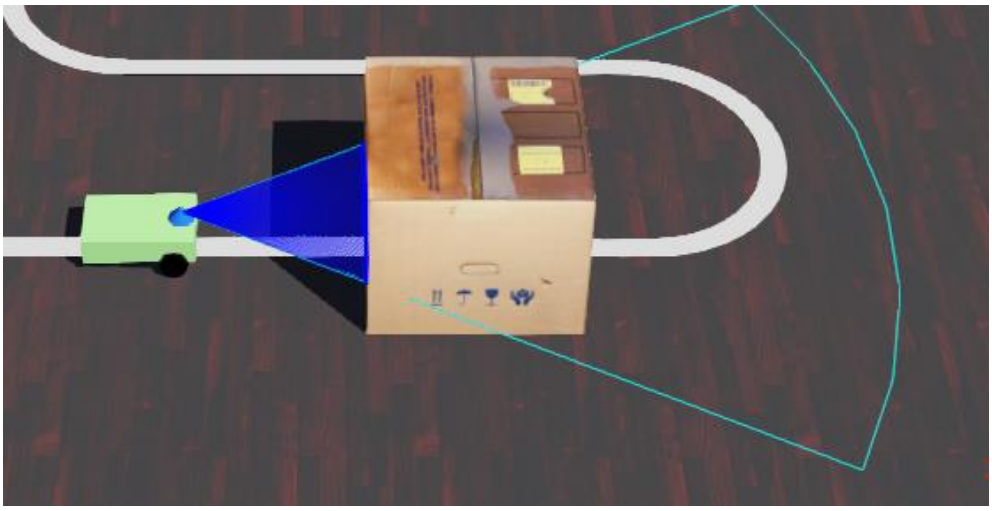

Рисунок 3.51 – Знаходження перешкоди

3.4 Додавання сенсору використання батареї робота

Для цього потрібно в налаштуваннях вузла робота знайти пункт battery. Далі, натиснувши правою кнопкою миші, створити три змінні-параметри (рис. 3.52).

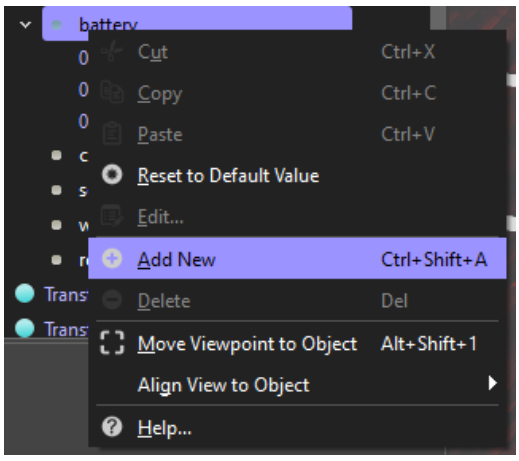

Рисунок 3.52 – Створення трьох змінних для батареї

Це поле має містити три значення: перше відповідає поточному рівню енергії робота в джоулях (Дж), друге – це максимальна енергія, яку може зберігати робот у джоулях, а третє – швидкість заряду енергії у ватах [Вт]=[Дж]/[с].

Симулятор оновлює перше значення, а два інших залишаються незмінними.

Коли поточне значення енергії досягає нуля, відповідний процес контролера припиняється, і змодельований робот припиняє весь рух.

Параметри, що були змодельовані, представлено на рис. 3.53.

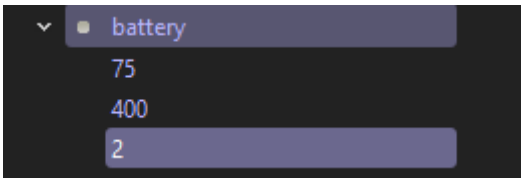

Рисунок 3.53 – Параметри моделювання батареї

Тут перший параметр – поточний параметр заряджання батареї, другий – максимальна потужність батареї, третій – параметр заряджання.

Також потрібно внести зміни в програму керування – контролер.

Потрібно зробити дозвіл на використання батареї:

robot.batterySensorEnable(timestep)

А також у циклі отримати поточне значення батареї:

print ("Battery lvl:", robot.batterySensorGetValue())

Результат виконання представлено на рис. 3.54.

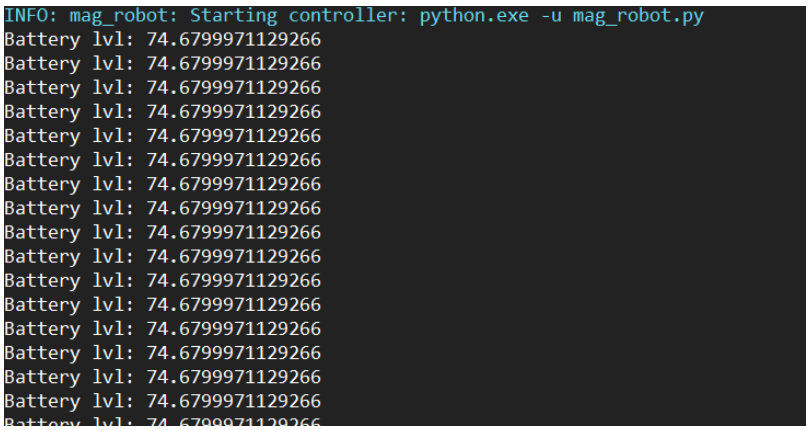

Рисунок 3.54 – Отримання даних батареї-акумулятору

Після того, як вичерпано заряд акумулятору, робот попереджає про те що розрядився (рис. 3.55).

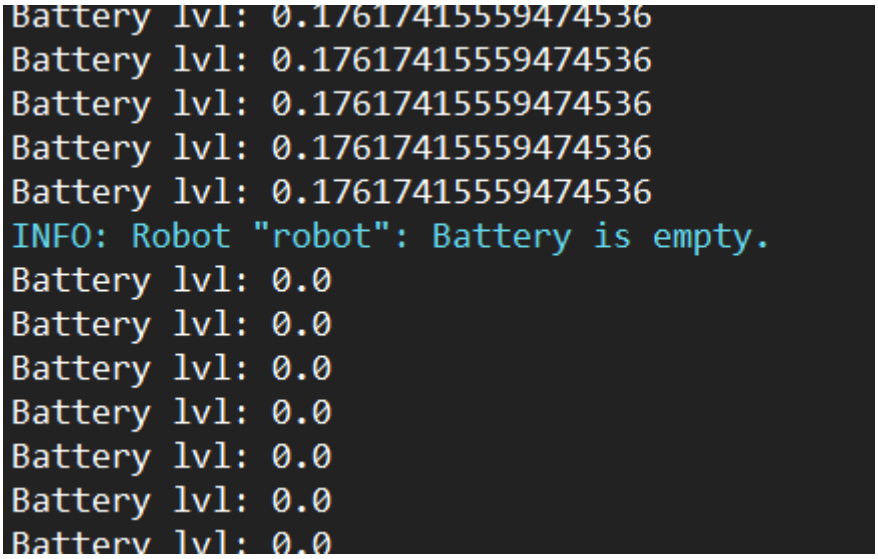

Рисунок 3.55 – Інформація про те що батарея розряджена

3.5 Додавання інфрачервоних сенсору

Було прийнято рішення для додавання ще й інфрачервоних сенсорів. Для цього було створено копію робота, з якого прибрали датчики лінії та лідар.

Далі було додано новий вузол DistanceSensor, до якого дочірнім елементом було обрано форму Shape. Геометрія форми – капсуль, висота –

0,01 м, радіус 0,015 м. Параметр appearance –PBRAppearance має наступні налаштування (рис. 3.56):

– baseColor – (1; 1; 1);

– roughtness – 0;

– metallness – 1.

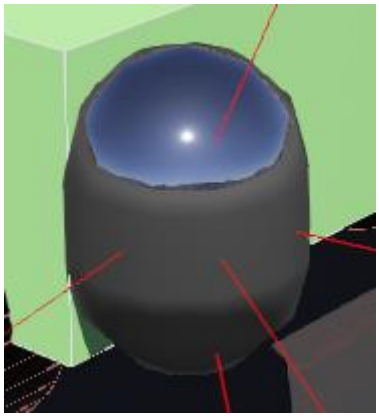

Рисунок 3.56 – Зовнішній вигляд інфрачервоного датчика

Далі задали наступні параметри (рис. 3.57):

– ім’я – distance_sensor;

– переміщення – (-0,04 м; -0,090032 м; 0 м);

– поворот за віссю z – -2,17 рад;

– пошукова таблиця – (0 м; 0 м.; 0 м) та (0,9 м; 0,9 м; 0 м) – максимальне значення, на якому спрацьовують датчики;

– тип – інфрачервоний;

– кількість променів – 5;

– апертура – 1,57 рад.

Для роботи майбутньої програми потрібно 2 датчики, тому було додано ще один зі зміненими параметрами переміщення та повороту – (0,04 м; -0,089968м; 0 м) та -0,97 рад відповідно.

Загальний вигяд робота представлено на рис. 3.58.

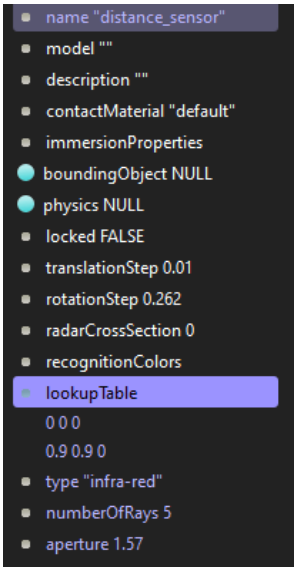

Рисунок 3.57 – Параметри датчика

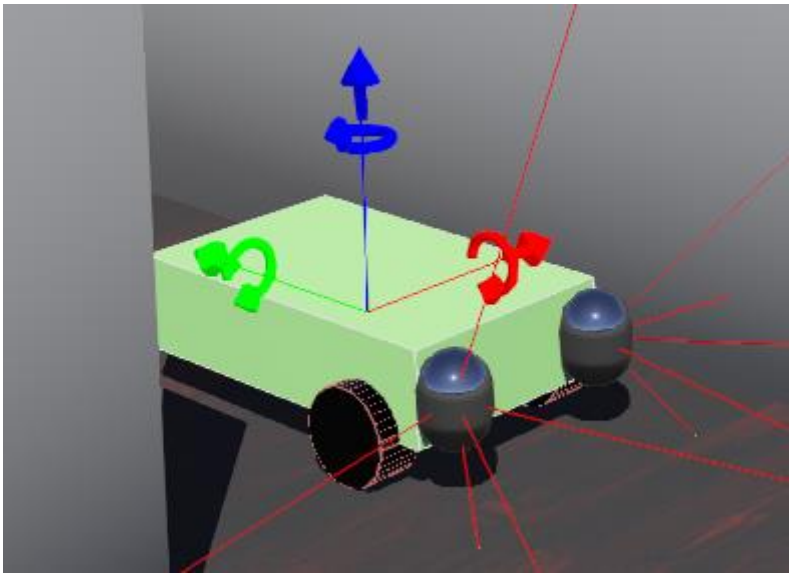

Рисунок 3.58 – Загальний вигляд робота з інфрачервоними датчиками

Для другого робота було змінено і програму керування (контролер).

По-перше, створили функцію для переміщення робота run_robot(robot).

Визначили обидва мотори, їх початкову швидкість та позицію:

```
left_motor = robot.getDevice('left_motor') # взято з моделі робота
    right_motor = robot.getDevice('right_motor')
    left_motor.setPosition(float('inf'))
    right_motor.setPosition(float('inf')) # inf = швидкість
```

```
left_motor.setVelocity(0.0) # початкова швидкість 0
right_motor.setVelocity(0.0)
```

Далі створили об’єкти сенсорів та дозволили використання в симуляторі:

```
distance_sensor = robot.getDevice('distance_sensor')
distance_sensor2 = robot.getDevice('distance_sensor2')
distance_sensor.enable(TIMESTEP)
distance_sensor2.enable(TIMESTEP)
```

У циклі while robot.step(TIMESTEP) != -1 визначили отримання даних з цих сенсорів:

```
print(distance_sensor.getValue(),distance_sensor2.getValue() )
```

Результати показані на рис. 3.59.

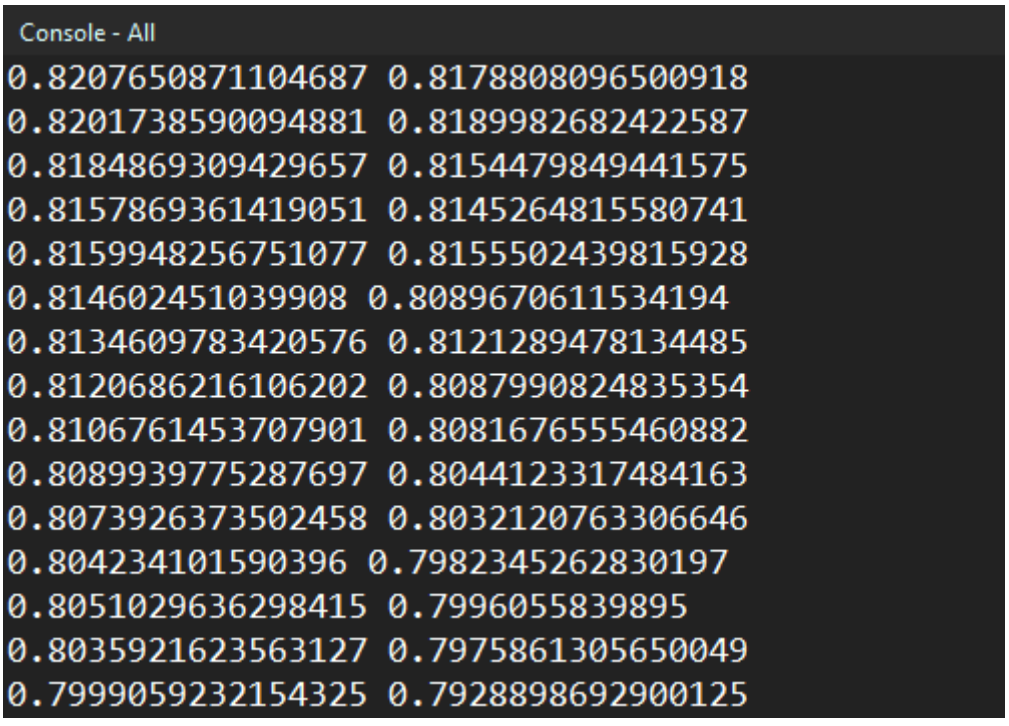

Рисунок 3.59 – Отримання результатів з датчиків

3.6 Висновки до розділу 3

В результаті виконання 3 розділу було вирішено наступні завдання:

– створено світ, в якому буде працювати віртуальна модель робота;

– створено модель робота у середовищі WeBots, що має два колеса з моторами та одне підтримуюче колесо;

– додано 8 ІЧ-датчиків, за допомогою яких робот може рухатись лінією;

– розроблено програмне забезпечення для переміщення робота;

– додано лідар та налаштовано для знаходження перешкод при русі по лінії;

– додано акумуляторну батарею для робота;

– додано ІЧ-датчики для проходження роботом невідомого приміщення.

4 ПРОВЕДЕННЯ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ СЕНСОРНОЇ СИСТЕМИ РОЗРОБЛЕНОГО РОБОТА

4.1 Експериментальні дослідження ПІД-регулятора робота та його налаштування для руху лінією

Програмне забезпечення для моделювання WeBots підтримує завантаження зовнішніх файлів з певними типами даних, тому було прийнято рішення розробити декілька треків для налаштування ПІД-регулятора.

Для створення цих треків було обрано середовище TinkerCad, що дозволяє створити моделі онлайн. Треки мають різні форми, серед яких:

– коло;

– прямокутник;

– складна траєкторія.

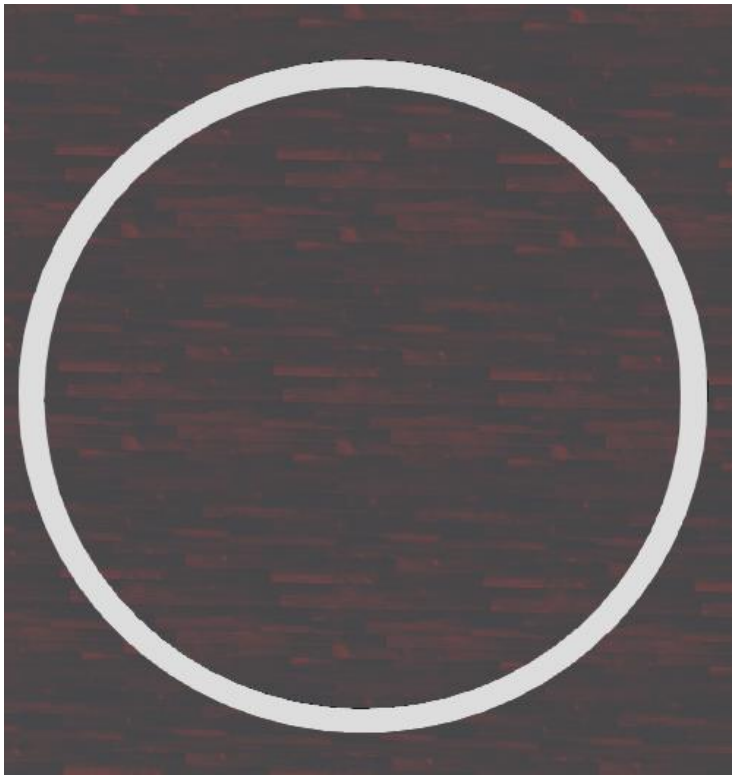

Рисунок 4.1 – Траєкторія коло для налаштування ПІД-регулятора

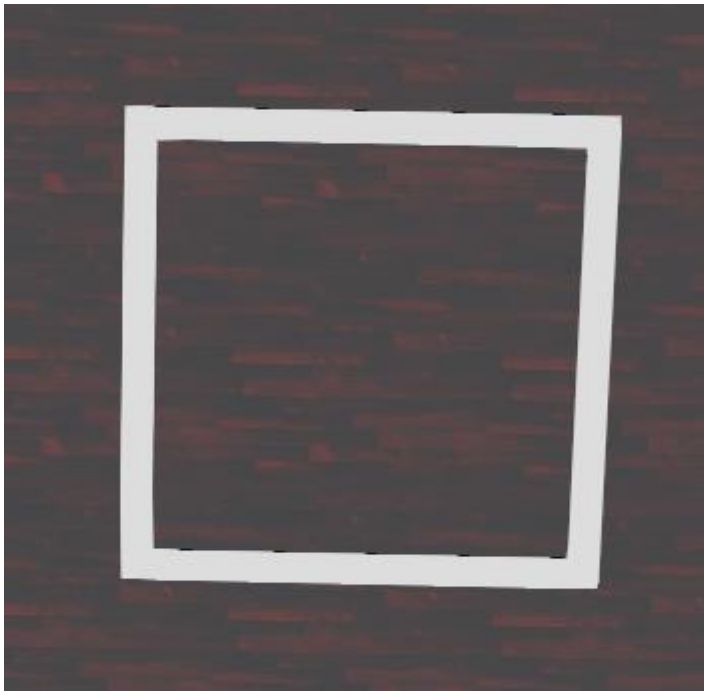

Рисунок 4.2 – Траєкторія прямокутник для налаштування ПІД-регулятора

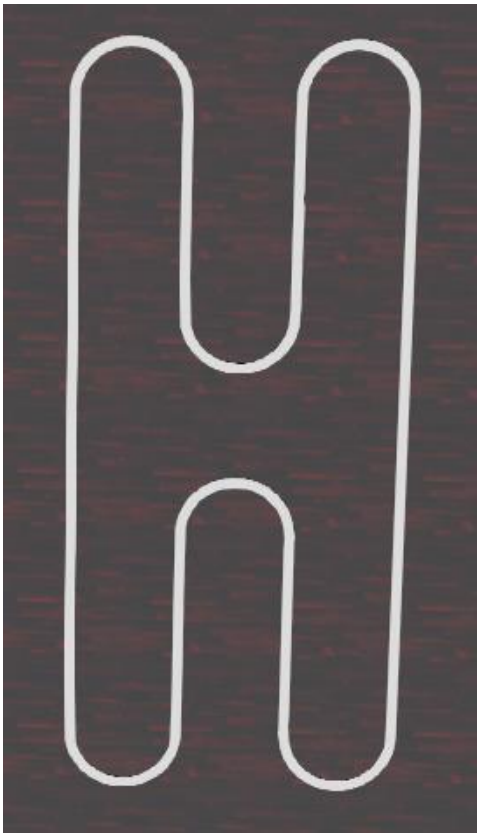

Рисунок 4.3 – Складна траєкторія для налаштування ПІД-регулятора

Для того щоб завантажити моделі в вузли WeBots, потрібно виконати дії, що будуть наведені нижче.

Першим кроком є додавання вузлу Transform, що підтримує масштабування завантажених файлів (рис. 4.4).

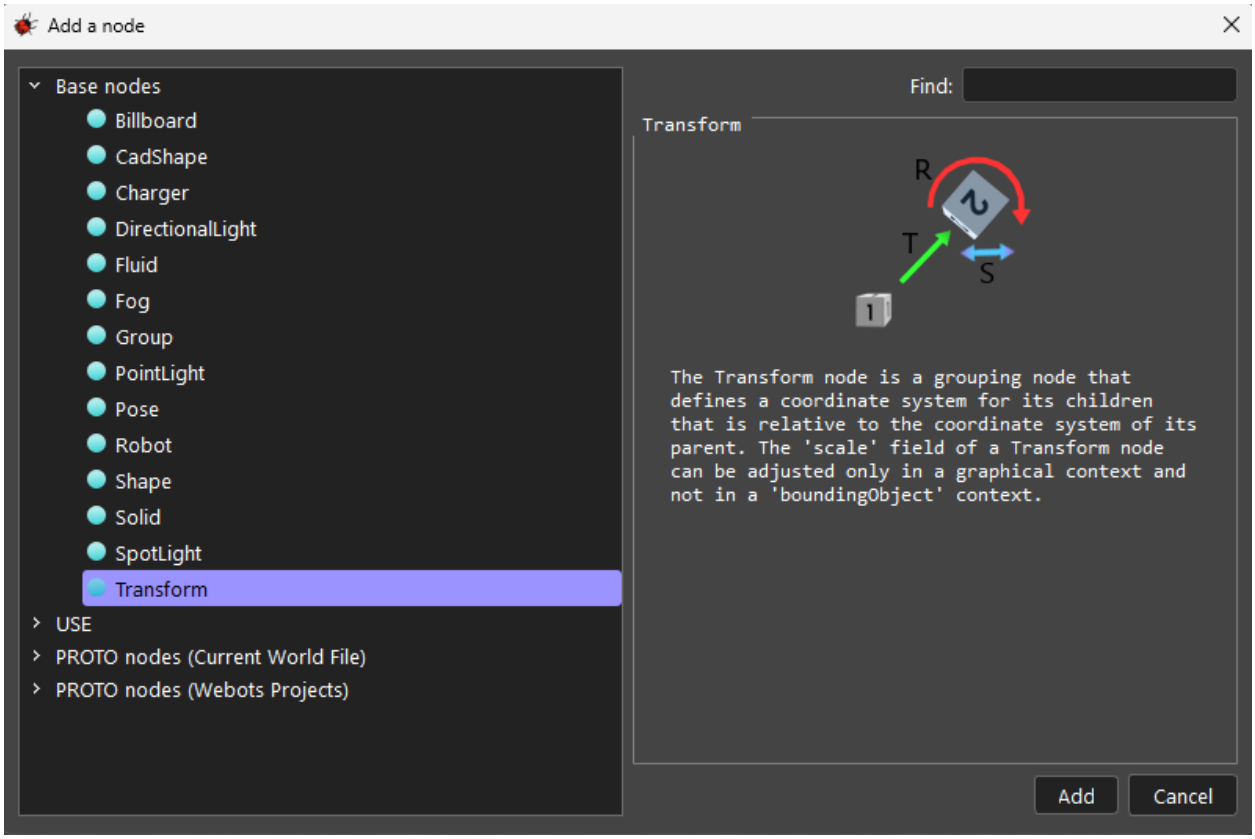

Рисунок 4.4 – Створення вузлу для завантаження файлів

Для цього вузлу створимо дочірній елемент Shape, в якості геометрії оберемо Mesh, що дозволяє завантажувати тривимірні об’єкти зі зовнішніх файлів (рис. 4.5).

В якості параметра url потрібно обрати зовнішні файли певних форматів (рис. 4.6), після чого ці об’єкти з’являться у створеному світі.

Так як буде використано три треки, то було створено і три файли для кожного з них.

Для кола було змінено параметри масштабування на (0,01 м; 0,01 м; 0,0001 м).

Для квадрату масштабування змінено на (0,02 м; 0,02 м; 0,001 м).

Для складної траєкторії – (0,02 м; 0,005 м; 0,02 м).

Найскладнішою ділянкою для налаштування та проходження роботом є гострі кути, тому квадратна траєкторія є найскладнішою.

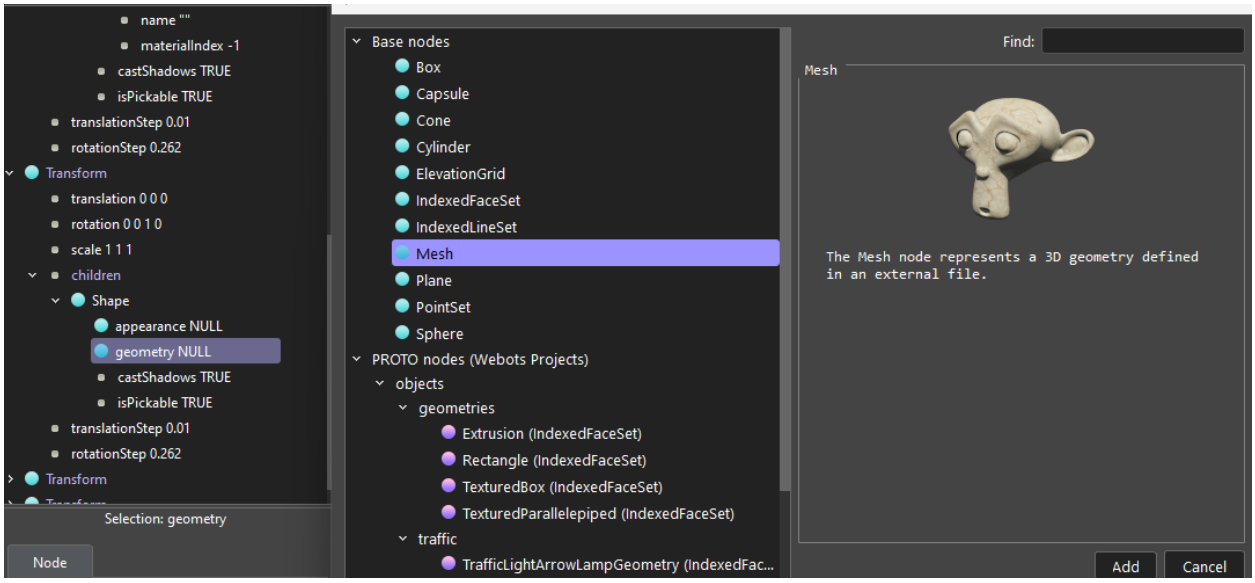

Рисунок 4.5 – Створення вузлу Mesh для додавання файлів

Рисунок 4.6 – Формати Mesh-файлів для додавання

Проведемо експериментальні дослідження з налаштування ПІД-регулятора для траєкторії коло. Результати налаштування показані в табл. 4.1.

Як бачимо з результатів, записаних в таблицю, при пропорційній складовій, що дорівнює 0,1, і значеннях диференційної від 0 до 0,9, робот не може пройти траєкторію коло.

Інтегральна складова змінює швидкості повороту робота дуже сильно, навіть при значенні 0,1, тому було прийнято рішення її обирати або дуже малою (менше 0,0001), або взагалі не використовувати.

Диференційна складова відповідає за більш точне налаштування поворотів вже у протилежний напрямок руху, або доналаштування робота на поворот.

Таблиця 4.1 – Експериментальні дослідження для кола

№	Kp	Ki	Kd	Виконання
1	0,1	0	0	ні
2	0,1	0	0,1	ні
3	0,1	0	0,2	ні
4	0,1	0	0,3	ні
5	0,1	0	0,4	ні
6	0,1	0	0,5	ні
7	0,1	0	0,6	ні
8	0,1	0	0,7	ні
9	0,1	0	0,8	ні
10	0,1	0	0,9	ні
11	0,2	0	0	так
12	0,2	0	0,1	так
13	0,2	0	0,2	так
14	0,2	0	0,3	так
15	0,2	0	0,4	так
16	0,2	0	0,5	так
17	0,2	0	0,6	так
18	0,2	0	0,7	так
19	0,2	0	0,8	так
20	0,2	0	0,9	так
21	0,3	0	0,1	так
22	0,3	0	0,9	так
23	0,4	0	0	так
24	0,5	0	0	так
25	0,6	0	0	так
26	0,7	0	0	так
27	0,8	0	0	так
28	0,9	0	0	так

Також при збільшенні пропорційної складової до 0,2 робот вже може пройти траєкторію коло, але не повороти відбуваються не дуже плавно. Так само і при зміні диференційної складової з 0 до 0,9 робот починає наче тремтіти під час повороту, тому краще використовувати середнє значенні цієї складової.

Під час проведення експерименту було виявлено, що найкращі показники для даного робота і траєкторії кола є значення (рис. 4.7):

$$kp = 0{,}5;\ ki = 0; kd = 0{,}2.$$

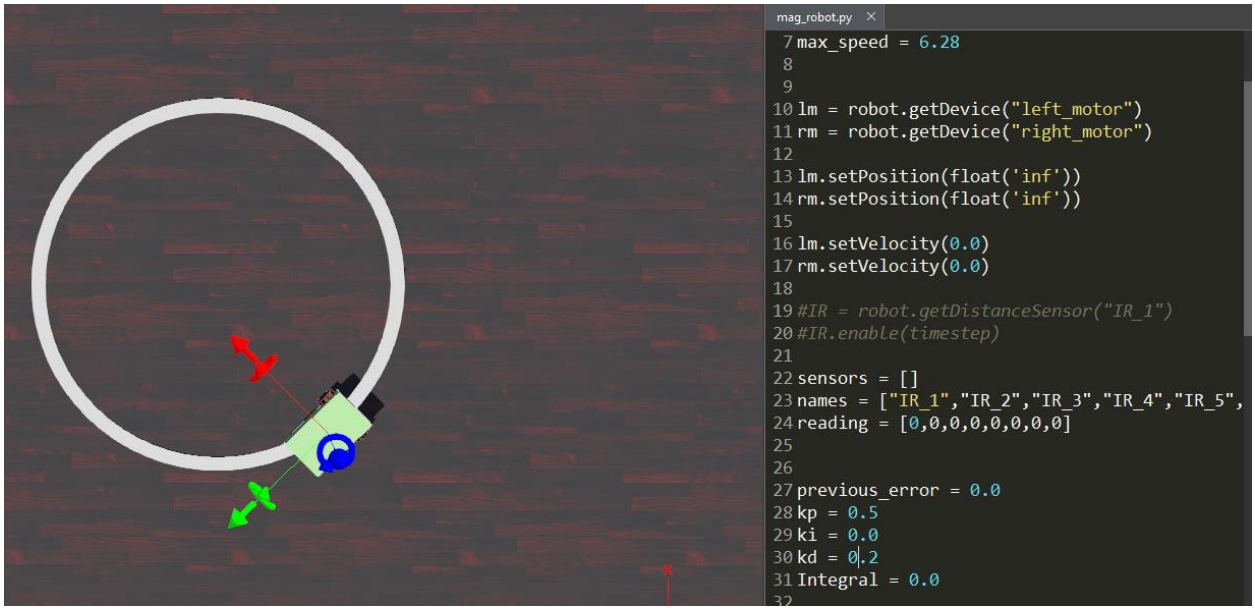

Рисунок 4.7 – Проходження траєкторії коло

Проведемо аналогічні експериментальні дослідження з налаштування ПІД-регулятора для траєкторії квадрат. Результати налаштування показані в табл. 4.2.

Як видно з результатів проведених експериментів, робот починає проходити траєкторію при пропорційному коєфіцієнті, що дорівнює 0,5, але його рух є дуже різким та ненадійним і будь-який момент робот може зійти з траєкторії. Додаванням диференційної складової його поведінка стала краще, але рухи все одно є достатньо ненадійними.

При збільшенні пропорційних і диференційних коєфіцієнтів поведінка і шлях робота стають більш плавними та надійними.

В результаті експерименту було виявлено що краще за все робот працює з налаштуваннями пропорційної, інтегральної та диференційної складових – (0,9; 0,0; 0,9) (рис. 4.8).

Таблиця 4.2 – Експериментальні дослідження для квадрата

№	Kp	Ki	Kd	Виконання
1	0,1	0	0	ні
2	0,1	0	0,5	ні
3	0,1	0	0,9	ні
4	0,2	0	0	ні
5	0,2	0	0,5	ні
6	0,2	0	0,9	ні
7	0,3	0	0	ні
8	0,3	0	0,5	ні
9	0,3	0	0,9	ні
10	0,4	0	0	ні
11	0,4	0	0,5	ні
12	0,4	0	0,9	ні
13	0,5	0	0	так
14	0,5	0	0,5	так
15	0,5	0	0,9	так
16	0,6	0	0	так
17	0,7	0	0	так
18	0,8	0	0	так
19	0,9	0	0	так
20	0,9	0	0,9	так

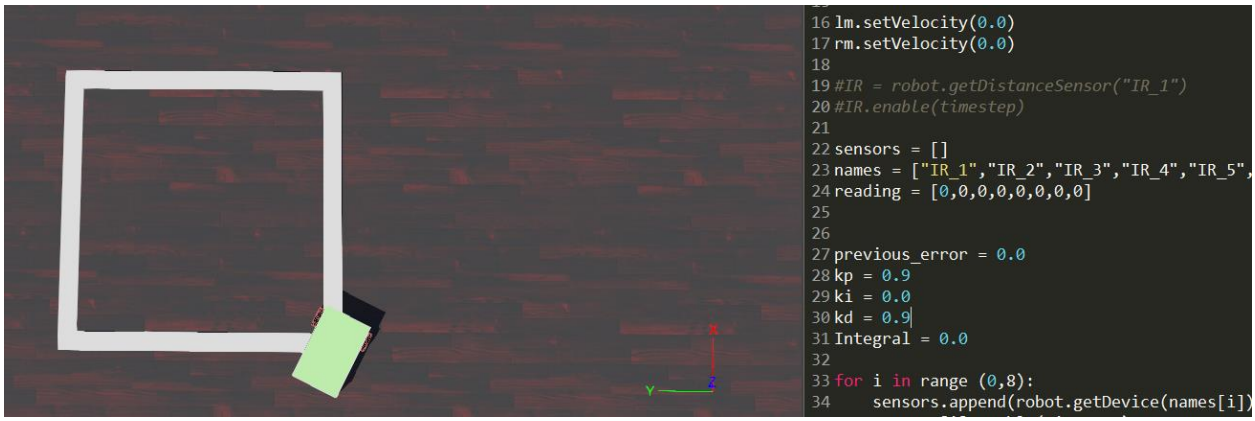

Рисунок 4.8 – Проходження траєкторії квадрат

Проведемо аналогічні експериментальні дослідження з налаштування ПІД-регулятора для складної траєкторії. Результати налаштування показані в табл. 4.3.

Таблиця 4.3 – Експериментальні дослідження для складної траєкторії

№	Kp	Ki	Kd	Виконання
1	0,1	0	0	ні
2	0,1	0	0,1	ні
3	0,1	0	0,2	ні
4	0,1	0	0,3	ні
5	0,1	0	0,4	ні
6	0,1	0	0,5	ні
7	0,1	0	0,6	ні
8	0,1	0	0,7	ні
9	0,1	0	0,8	ні
10	0,1	0	0,9	ні
11	0,2	0	0	так
12	0,2	0	0,1	так
13	0,2	0	0,2	так
14	0,2	0	0,3	так
15	0,2	0	0,4	так
16	0,2	0	0,5	так
17	0,2	0	0,6	так
18	0,2	0	0,7	так
19	0,2	0	0,8	так
20	0,2	0	0,9	так
21	0,3	0	0,1	так
22	0,3	0	0,9	так
23	0,4	0	0	так
24	0,5	0	0	так
25	0,6	0	0	так
26	0,6	0	0.3	так
26	0,7	0	0	так
27	0,8	0	0	так
28	0,9	0	0	так
29	0,9	0	0,5	так
30	0,9	0	0,9	так

Як видно з таблиці, робот починає проходити траєкторію з пропорційною складовою 0,2, але його траєкторія є дуже нестійкою та поведінка робота є різкою.

Експериментальним шляхом було визначено, що оптимальні параметри ПІД-регулятора – пропорційна, інтегральна та диференційна дорівнюють (0,6; 0.0; 0,3) відповідно.

Рисунок 4.9 – Проходження складної траєкторії

4.2 Проведення експериментів з інфрачервоними датчиками

В якості лабіринту для проходження роботом було обрано файл лабіринту, розроблений у TinkerCad.

Було створено новий вузол Transform (рис. 4.10), у якості дочірнього елементу обрано Shape, геометрія – Mesh.

Основні параметри лабіринту:

– переміщення – (-0,111 м; 3,83 м; -0,0616 м);

– поворот за віссю z – 1,57 рад;

– масштаб за осями – (0,03; 0,03; 0,03).

Параметри зовнішнього вигляду представлено на рис. 4.11.

Цей лабіринт було розташовано поруч з попередньою моделлю (рис. 4.12).

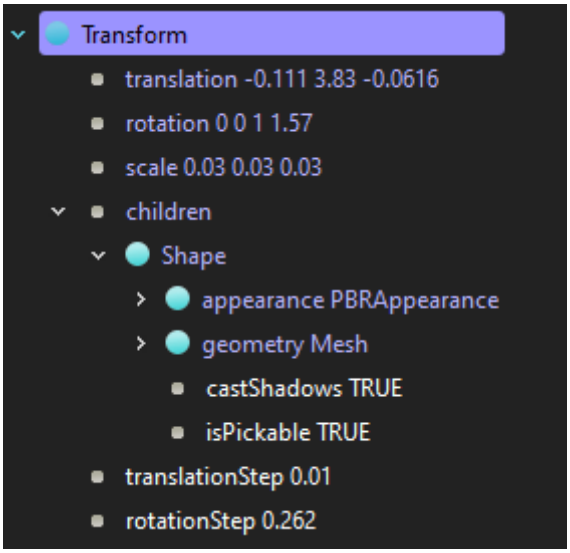

Рисунок 4.10 – Параметри переміщення доданої моделі лабіринту

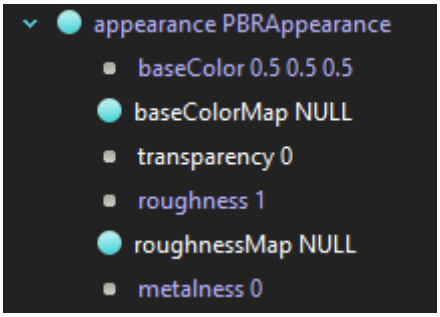

Рисунок 4.11 – Параметри зовнішнього вигляду лабіринту

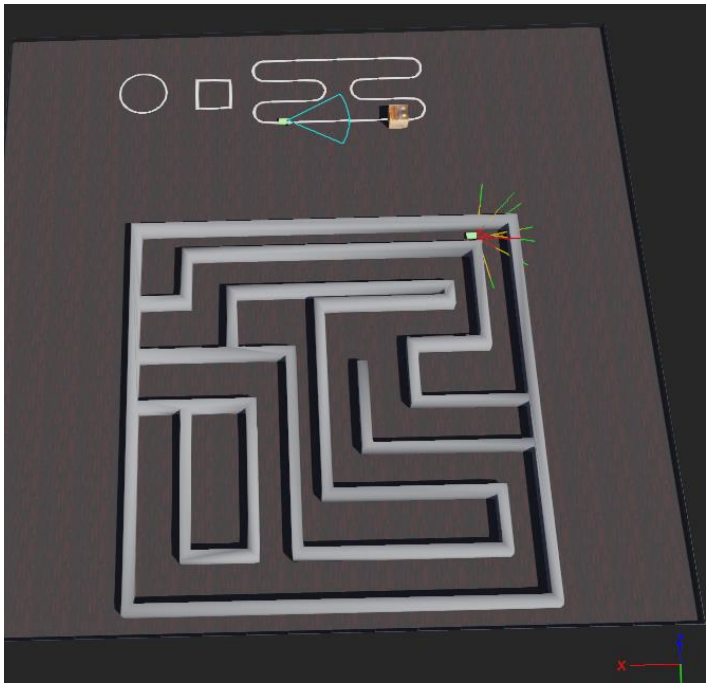

Рисунок 4.12 – Розташування елементів моделювання

Для проходження лабіринту також було створено новий контролер (рис. 4.13) під назвою maze-cont.

mag_robot.py | maze_contr.py

```
from controller import Robot

TIMESTEP = 32
MAX_SPEED = 6.28


def  run_robot(robot):

    # екземпляри моторів

    left_motor = robot.getDevice('left_motor') # взято з мо
    right_motor = robot.getDevice('right_motor')

    left_motor.setPosition(float('inf'))
    right_motor.setPosition(float('inf')) # inf = швидкість

    left_motor.setVelocity(0.0) # початкова швидкість 0
    right_motor.setVelocity(0.0)

    distance_sensor = robot.getDevice('distance_sensor')
    distance_sensor2 = robot.getDevice('distance_sensor2')
    distance_sensor.enable(TIMESTEP)
    distance_sensor2.enable(TIMESTEP)



    while robot.step(TIMESTEP) != -1:
       if distance_sensor.getValue()>0.6 and distance_senso
           left_motor.setVelocity(3.14) # початкова швидкіс
           right_motor.setVelocity(3.14)
       elif distance_sensor.getValue()<0.6:
           left_motor.setVelocity(0) # початкова швидкість
           right_motor.setVelocity(3.14)
       elif distance_sensor2.getValue()<0.6:
           left_motor.setVelocity(3.14) # початкова швидкіс
           right_motor.setVelocity(0)
```

Рисунок 4.13 – Новий контролер для робота

Для проходження лабіринту експериментальним шляхом було підібрано коефіцієнти та додано код з умовами:

```
if distance_sensor.getValue()>0.6 and distance_sensor2.getValue()>0.6:
        left_motor.setVelocity(3.14) # початкова швидкість 0
        right_motor.setVelocity(3.14)
    elif distance_sensor.getValue()<0.6:
        left_motor.setVelocity(0) # початкова швидкість 0
```

```
        right_motor.setVelocity(3.14)
    elif distance_sensor2.getValue()<0.6:
        left_motor.setVelocity(3.14) # початкова швидкість 0
        right_motor.setVelocity(0)
```

Цей код дозволив робот подолати лабіринт. Якщо відстань між датчиками робота та перешкодою більше 0,6 м, то робот їде прямо, якщо показання з лівого датчика менше за 0,6 м, то робот зупиняє лівий мотор і повертає праворуч, якщо показання з правого датчика менше за 0,6 м, то робот зупиняє правий мотор і повертає ліворуч.

Ці налаштування були зроблені саме для поточного лабіринту і відповідними розмірами робота та відстаней між стінками лабіринту.

4.3 Загальні правила безпеки роботи у приміщенні з комп’ютером

Розроблення симуляції відбувається в приміщенні де встановлені комп’ютери. Шкідливих факторів роботи за комп'ютером є чимало. Усі вони різною мірою впливають наш організм. Серед таких факторів виділяють:

– нагрівання деталей та поверхні комп'ютера в процесі активної експлуатації, що призводить до підвищення рівня температури у приміщенні;

– високе зорове навантаження при тривалій роботі;

– монотонність робочого процесу.

– ризик ураження електричним струмом у разі неправильної експлуатації або ігнорування фактів поломки техніки.

– високий рівень контрасту та блиску екрану, що згубно впливає на зір працівника.

Можна виділити вимоги для безпеки праці роботи за комп'ютером:

– дотримання загальних правил використання комп'ютерної техніки;

– дотримання мікроклімату приміщення, в якому проводиться робота;

– дотримання рівня шуму, що походить від комп'ютерної техніки;

– дотримання рівня освітлення під час роботи з комп'ютерною технікою;

– дотримання норм електромагнітних полів.

– проходження медичної комісії працівниками, діяльність яких пов'язана із регулярною роботою за комп'ютером;

Важливо дотримуватись і заходів безпеки під час роботи за комп'ютером:

– не чіпати комп'ютер та системний блок мокрими руками.

– не класти на корпус комп'ютера сторонні предмети, самостійно чистити системний блок від пилу та бруду, за умови його підключення до напруги;

Якщо ви виявлено неполадки в роботі комп'ютера – негайно припинити роботу за ним та повідомити про це керівника чи відповідні служби.

– не вмикати або вимикати комп'ютер із розетки без особливої потреби та часто під час робочого процесу.

Після завершення роботи за комп'ютером працівник забов’язаний вимкнути комп'ютер із мережі, дотримуючись усіх правил безпеки.

Всі перелічені заходи безпеки допоможуть максимально ефективно використовувати комп'ютер у робочій діяльності і не завдати шкоди своєму здоров'ю та здоров'ю колег. [33]

4.4 Висновки до розділу 4

В результаті виконання 4 розділу було проведено експериментальні дослідження з розробленим роботом, а саме:

– у середовищі TinkerCad було створено траєкторії для проведення експериментальних досліджень;

– було налаштовано ПІД-регулятор для проходження роботом різних траєкторій, серед яких рух по колу, рух по квадрату, а також по складній траєкторії. Для руху по колу, коефіцієнти ПІД-регулятора склали: kp =

0,5; ki = 0; kd = 0,2; для руху по квадрату – (0,9; 0,0; 0,9); для руху складною траєкторією – (0,6; 0.0; 0,3);

– додано другого робота для проходження лабіринту;

– додано лабіринт для проходження роботом, а також експериментально було підібрано коєфіцієнти для ІЧ-датчиків, щоб робот міг пройти цей лабіринт.

ВИСНОВКИ

В результаті виконання магістерської кваліфікаційної роботи всі поставлені завдання, а саме підвищення керування роботизованих систем за рахунок розроблення їх комп’ютерної моделі, шляхом розробки симуляції роботи робота та його сенсорної системи.

Перший розділ демонструє аналіз сучасного стану систем моделювання у Індустрії 5.0, зокрема етапів моделювання, а також його використання в робототехніці, а також виділено, що комп’ютерне моделювання має ряд своїх завдань для розвитку сучасної робототехніки за рахунок побудови більш дешевих комп’ютерних моделей, ніж реальні прототипи.

В результаті виконання 2 розділу було проведено порівняння різних середовищ моделювання роботи роботів, таких як WeBots, Gazebo, CoppeliaSim та MATLAB/Simulink, обрано саме WeBots завдяки можливості провести в ньому моделювання власної моделі робота та зручного моделювання датчиків, розглянуто створення простого світу для моделювання робота, а також додавання інших елементів оточення, додаючи цим елементам фізичних властивостей, таких як силу тяжіння та масу, а також розглянуто питання теорії автоматичного управління та знайдено передавальну функцію двигуна і перевірено його стійкість.

В результаті виконання 3 розділу було створено світ, в якому буде працювати віртуальна модель робота, створено самого робота у середовищі WeBots, додано до нього 8 ІЧ-датчиків, за допомогою яких робот може рухатись лінією, розроблено програмне забезпечення для переміщення робота. Далі було додано лідар та налаштовано для знаходження перешкод при русі по лінії, а також акумуляторну батарею для робота і ІЧ-датчики для проходження роботом невідомого приміщення.

В результаті виконання 4 розділу було проведено експериментальні дослідження з розробленим роботом – налаштовано ПІД-регулятор для

проходження роботом різних траєкторій, серед яких рух по колу, рух по квадрату, а також по складній траєкторії. Для руху по колу, коефіцієнти ПІД-регулятора склали $\mathrm{kp} = 0{,}5;\ \mathrm{ki} = 0;\ \mathrm{kd} = 0{,}2$, для руху по квадрату – (0,9; 0,0; 0,9), для руху складною траєкторією – (0,6; 0.0; 0,3). Також додано другого робота для проходження лабіринту і експериментально було підібрано коефіцієнти для ІЧ-датчиків, щоб робот міг його пройти.

Результати роботи можна віднести до Цілей сталого розвитку 9 та 12: «Промисловість, інновації та інфраструктураа», «Відповідальне споживання та виробництво» відповідно [32].

Отримані результати можна використовувати на етапах проектування роботизованих систем, шляхом використання математичних моделей симуляції роботи робота та його сенсорної системи, для тестування та наладки, а також в освітньому процесі при проведенні практичних чи лабораторних робіт за спеціальністю 174 Автоматизація, комп’ютерно-інтегровані технології та робототехніка.

Результати, що було отримано під час навчання та підготовки кваліфікаційної роботи висвітлено в статті [3]: Buts D. Signals Collisions Detection In Wireless Networks / D. Buts, O. Chala, S. Maksymova // Journal of Universal Science Research. – 2023. – № 1(11). – P. 156–168. (Додаток А).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008:2015 Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. ДП «УкрНДНЦ», 2016. – 31 с.

2. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка, освітньо-професійних програм: «Комп'ютерно-інтегровані технологічні процеси і виробництва», «Комп'ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, Р. В. Артюх, В. В. Безкоровайний, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал. Харків: ХНУРЕ, 2023. 49 с.

3. Buts D. Signals Collisions Detection In Wireless Networks / D. Buts, O. Chala, S. Maksymova // Journal of Universal Science Research. – 2023. – № 1(11). – P. 156–168.

4. Невлюдов І. Ш., Андрусевич А. О., Євсєєв В. В., Новоселов С. П., Демська Н. П. Проектування мобільних маніпуляційних роботів: Монографія. – Х. :, 2022. – 427 с.

5. He, M. Industry 5.0, Future of Workforce Beyond Efficiency and Productivity. / M.He, , Bun Chand, // Innovation, Sustainability, and Technological Megatrends in the Face of Uncertainties. Future of Business and Finance. Springer, Cham, 2024, 21 p.

6. Adel, A.. Future of industry 5.0 in society: Human-centric solutions, challenges and prospective research areas / A. Adel // J Cloud Comp, vol 11, iss. 40, 2022, 17 p.

7. Garau, C. Industry 5.0: Opportunities and challenges for small and medium enterprises / Garau, C., & Vanino, E // International Journal of Production Research, vol. 58, iss. 2, 2020, pp. 579–592.

8. Palazhchenko, Y Industry 5.0: Aspects of Collaboration Technologies / Y.Palazhchenko, V.Shendryk, V. Ivanov, M.Hatala, // Flexible Automation and

Intelligent Manufacturing: Establishing Bridges for More Sustainable Manufacturing Systems. FAIM 2023, Volume 2, 2023, 17 p.

9. Ciucu-Durnoi A.N. Beyond Industry 4.0: Tracing the Path to Industry 5.0 through Bibliometric Analysis. / A.N. Ciucu-Durnoi, C. Delcea, A. Stănescu, C.A. Teodorescu, V.M. Vargas, // Sustainability, vol. 16, 2024, 58 p.

10. Yitmen I. Synopsis of Industry 5.0 Paradigm for Human-Robot Collaboration / I. Yitmen, A. Almusaed // Artificial Intelligence. IntechOpen, 2024, 15 p.

11. Mittal, S. Providing a User Extensible Service-Enabled Multi-Fidelity Hybrid Cloud-Deployable SoS Test and Evaluation (T&E) Infrastructure: Application of Modeling and Simulation (M&S) as a Service (MSaaS) / Mittal, S.; R.L. Wittman, J. Gibson, J. Huffman, H. Miller // Information 2023, vol 14, 26 p.

12. Gera, R. A Comprehensive and Narrative Review of Industry 5.0 Technologies: 2018–2022 / R. Gera, P. Chadha, G.S. Khera, R. Yadav // Renewable Energy Optimization, Planning and Control. Studies in Infrastructure and Control. Springer, Singapore, 2023, pp. 237-259.

13. Britannica, The Editors of Encyclopaedia. Computer simulation [Електронний ресурс]. – Режим доступу: https://www.britannica.com/technology/computer-simulation.

14. Cotta, W.A.A. The Cognitive Factory in Industry 5.0: From Concept to Implementation / W.A.A.Cotta, S.I. Lopes, R.F. Vassallo // Smart Cities 2023, vol. 6, 2023, pp. 1901-1921.

15. Ortega Sainz Argentina. Composable and Executable Scenarios for Simulation-Based Testing of Mobile Robots / Argentina Ortega Sainz, Samuel Parra, Sven Schneider, Nico Hochgeschwender // Frontiers in Robotics and AI, vol. 11, 2024, 18 p.

16. Mittal, Mayank and others / Orbit: A Unified Simulation Framework for Interactive Robot Learning Environments // IEEE Robotics and Automation Letters, vol. 8, iss 6, 2023, pp. 3740–3747.

17. 3 Advantages of Using a Robot Simulator | GBC Robotics (onlinerobotics.com) [Електронний ресурс]. – Режим доступу: https://www.onlinerobotics.com/news-blog/3-advantages-using-robot-simulator.

18. Hanna, J.P., Desai, S., Karnan, H. Grounded action transformation for sim-to-real reinforcement learning. / J.P.Hanna, S. Desai, H. Karnan// Mach Learn, vol. 110, 2021, pp/ 2469–2499.

19. Choi HeeSun. On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward / HeeSun Choi, Cindy Crump, Christian Duriez, Asher Elmquist, Gregory Hager // Proceedings of the National Academy of Sciences of the United States of America, vol. 118 iss. 1, 2021, 13 p.

20. What Is Robot Simulation? - MATLAB & Simulink (mathworks.com) [Електронний ресурс]. – Режим доступу: https://www.mathworks.com/discovery/robot-simulation.html.

21. Amazing benefits of combining Ar, Vr and Robotics [Електронний ресурс]. – Режим доступу: https://www.roboticstomorrow.com/story/2022/06/5-amazing-benefits-of-combining-ar-vr-and-robotics/19052.

22 Webots User Guide [Електронний ресурс]. – Режим доступу: https://cyberbotics.com/doc/guide/index.

23 Gazebosim [Електронний ресурс]. – Режим доступу: https://gazebosim.org/docs/latest/getstarted.

24 Coppeliarobotics [Електронний ресурс]. – Режим доступу: https://www.coppeliarobotics.com.

25 Mathworks MatLab [Електронний ресурс]. – Режим доступу: https://www.mathworks.com/products/simulink.html.

26 First Simulation[Електронний ресурс]. – Режим доступу: webots/docs/guide/tutorial-1-your-first-simulation-in-webots.md.

27 Gazebo [Електронний ресурс]. – Режим доступу: https://roboticsimulationservices.com/ros-gazebo-everything-you-need-to-know/

28 CoppeliaSim Simulator [Електронний ресурс]. – Режим доступу: https://hades.mech.northwestern.edu/index.php/Getting_Started_with_the_CoppeliaSim_Simulator.

29 What is Matlab Simulink [Електронний ресурс]. – Режим доступу: https://www.simplilearn.com/tutorials/matlab-tutorial/what-is-matlab-simulink.

30. Теорія автоматичного управління (збірник задач) [Текст]: навч. посіб. Для спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології / І.Ш. Невлюдов, О.В. Токарєва; Харків. нац. ун-т радіоелектроніки. -Харків: Панов А.М., 2020. – 240 с.

31. Романенко, В. Д. Теорія керування і прогнозування у складних системах. Підручник [Електронний ресурс] : підручник для здобувачів ступеня магістра за спеціальністю «Системний аналіз» / В. Д. Романенко, Ю. Л. Мілявський ; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 6,17 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2024. – 404 с. – Назва з екрана. URI https://ela.kpi.ua/handle/123456789/67937. Дата доступу 27.12.2024.

32. Основи наукових досліджень : підручник / І. Ш. Невлюдов, Ю. М. Олександров, А. О. Андрусевич, О. О. Чала ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. – Prague : OKTAN PRINT, 2024. – 468 с.

33. Основи охорони праці та безпеки життєдіяльності : навч. посіб. / МОН України, Уманський держ. пед. ун-т імені Павла Тичини ; уклад. Н. В. Баличева. – Умань : Візаві, 2023. – 273 с.